Lecture 12

NP-Completeness

Problems that are **Easy (Class P)**

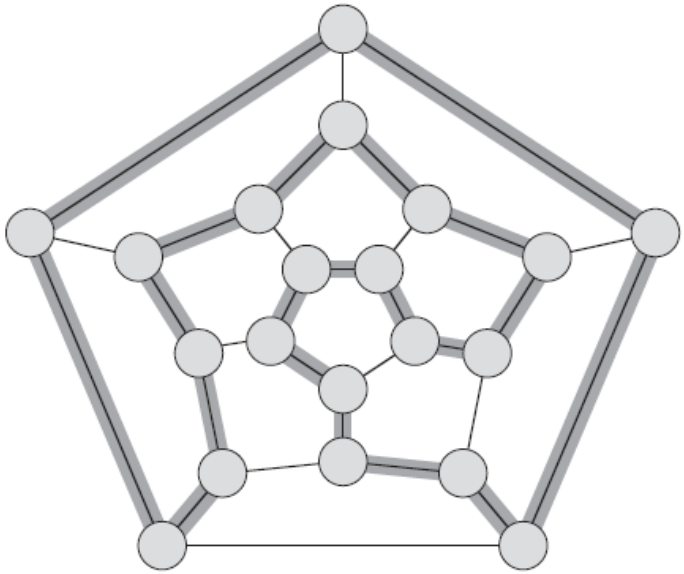
- Problems that can be solved by our computers (Turing Machines) in polynomial time.
- That means, problems for which there are **good algorithms**. Good algorithms means **polynomial time** algorithms.
- **Polynomial time** means, in $O(n^k)$ time, like $O(n^2)$, $O(n\log n)$, $O(n^3)$ time.
- Example: Sorting ($O(n \log n)$), Shortest Path by Dijkstra ($O(E \log n)$), LCS by Dynamic Programming ($O(mn)$).
- These problems are called in **Class P.**

Some Problems are **Difficult**

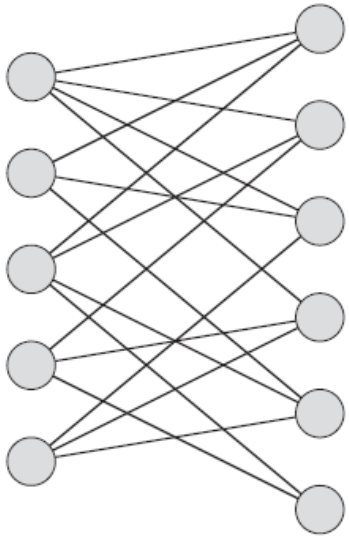
- Some problems can not be solved by our computers, that means by Turing Machines (laptops, desktops, mobiles, supercomputers).
- Why? Because, there are no **good algorithms** for them
- Which algorithms are **good algorithms**?
- **Answer:** Algorithms that have polynomial running time. That means $O(n^k)$. For example, $O(n^2)$, $O(n \log n)$, $O(n^3)$. Not exponential. That means not like $O(2^n)$.

Examples of Difficult Problems: Hamiltonian Cycle

- There are many problems that are hard:
- Example 1: Find **Hamiltonian Cycle** in a Graph
  - **Hamiltonian Cycle:** A cycle with every vertex only one time.



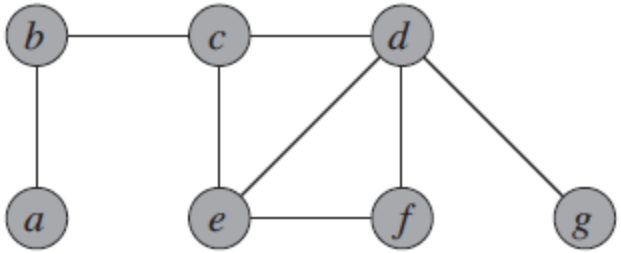
This graph has a Hamiltonian Cycle

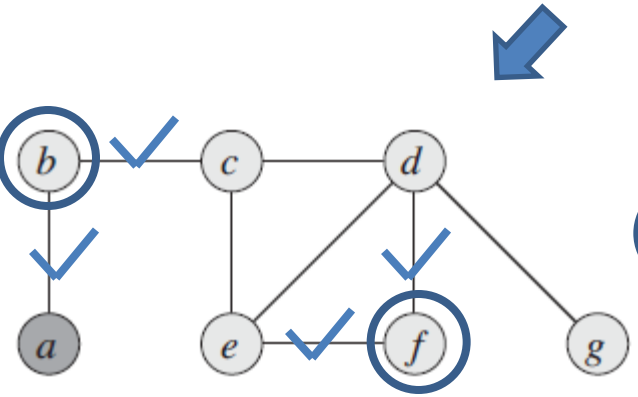


No Hamiltonian Cycle in this graph

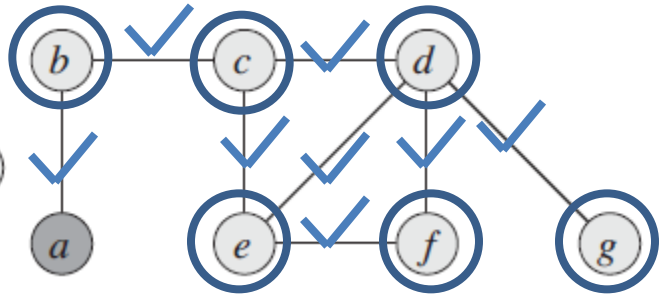
Examples of Difficult Problems: Vertex Cover

- Example 2: Find **Minimum Vertex Cover** in a graph.
  - **Vertex Cover:** set of vertices that cover all edges
  - **Minimum/k-size Vertex Cover:** vertex cover with minimum/k number of vertices.

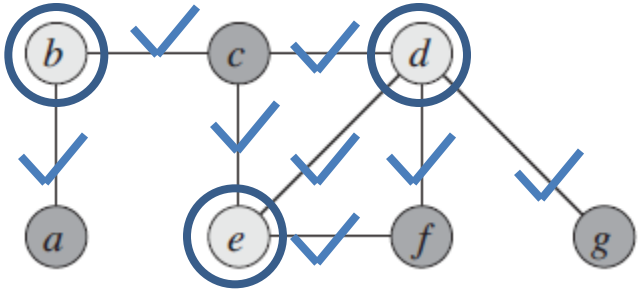




Vertex cover: {b,f} = NO



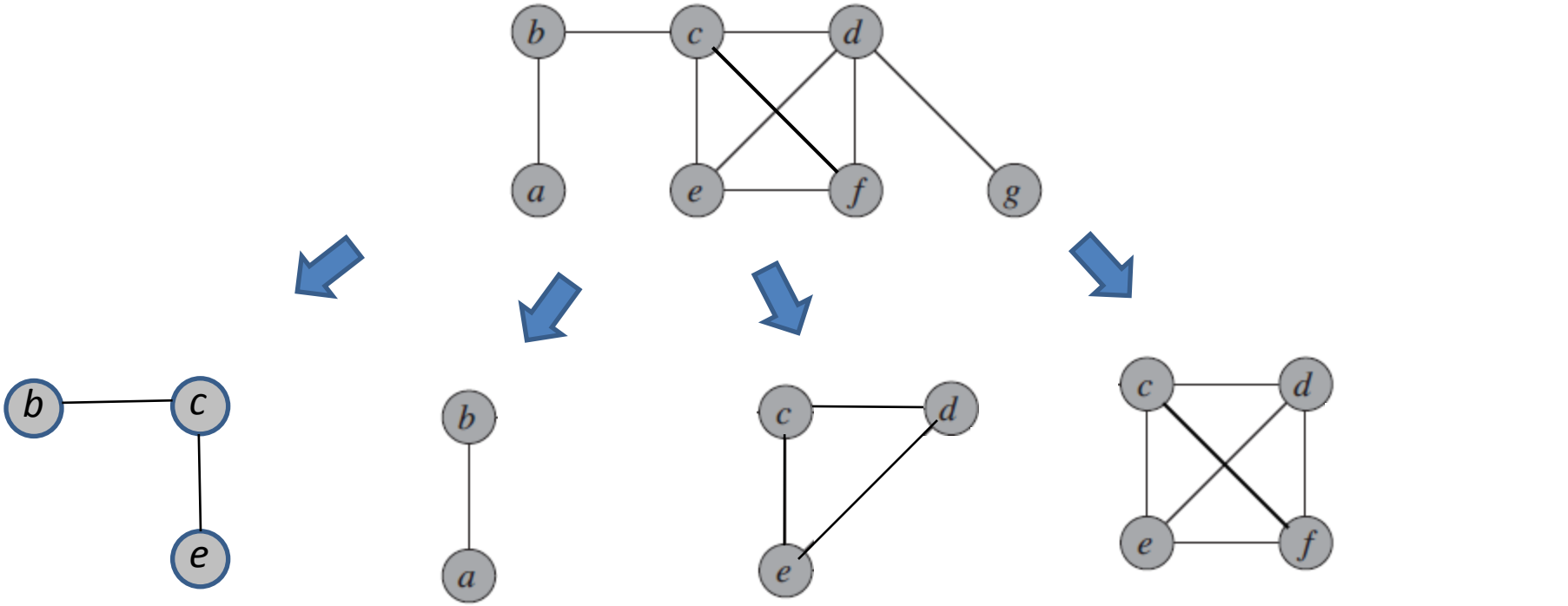
Vertex cover: {b,c,d,e,f,g} = YES = 6 vertices



Vertex cover: {b,d,e} = YES = 3 vertices = minimum

Examples of Difficult Problems: Clique

- Example 2: Find **Maximum Clique** in a graph. Or find a clique of size k.
  - **Clique:** Complete subgraph
  - **Minimum/k-size Clique:** Clique with Maximum/or k number of vertices.



Clique: {b,c,e} = NO

Clique: {b,c} = YES = 2 vertices

Clique: {c,d,e} = YES = 3 vertices

Clique: {c,d,e,f} = YES = 4 vertices = maximum

Examples of Difficult Problems: 3-SAT

- Example 3: **Circuit Satisfiability** (Easy version: **3-SAT**).
  - **3-SAT:** Given a Boolean formula in 3-conjunctive form, find an assignment of 0/1 to the variables, so that the result is 1

$(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

a	b	c	result
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

Many results 1

Example 1

$(a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge$
$(\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

a	b	c	result
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

No result 1

Example 2

Exponential Time Solutions for Difficult Problems

- Hard problems have exponential time algorithms for our computers: Compute all possible solution by brute–force method. Then check which solution is correct in polynomial time.
- Example: 3-SAT
  - There are n=3 variables a,b,c
  - There are $2^3 = 2^n = 8$ possible solutions
  - Try all of the one after another
  - For each one check whether it gives 0 or 1
  - Checking one solution takes polynomial time
  - For example: For a=0 b=1 c=0, the solution is
    $(0 \vee 1 \vee 0) \wedge (\neg 0 \vee \neg 1 \vee \neg 0) = (0 \vee 1 \vee 0) \wedge (1 \vee 0 \vee 1)$
    $= 1 \wedge 1 = 1$. So, this gives 1.
  - Time taken by this checking is: 3+3+3+3+1<5*3=5*n=O(n)
  - So, total time for checking all solutions = $2^3 * O(3) = 2^n * O(n)$ = Exponential.

$(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

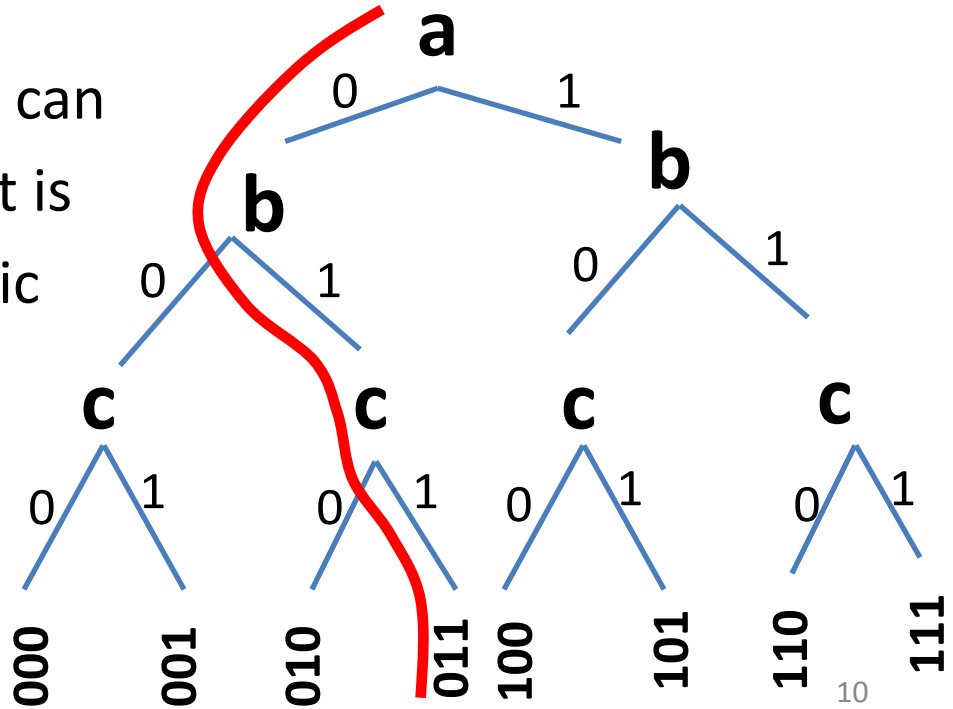
a	b	c	result
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

What Happens If Almighty Allah Gives a Special Computer?

- Special Computer means: **non-deterministic** computes. Our computers are called **deterministic**.
- These computers are not available in reality. We can only imagine.
- They can compute all possible solutions parallelly in polynomial time.
- So, total time for them is polynomial, not exponential
- For example: for 3-SAT problem it can generate all $2^3 = 2^n$ solutions in polynomial time and then can check whether the solutions are correct in polynomial time. So, total time polynomial*polynomial = polynomial.
- So, 3-SAT is exponential in deterministic computer, but **P**olynomial in **N**o-deterministic computer.
- 3-SAT is called NP (N from here, P from here) problem.
- Similarly, Vertex Cover, Hamiltonian Cycle are also in NP Class.

Compare Deterministic and Non-Deterministic Computers

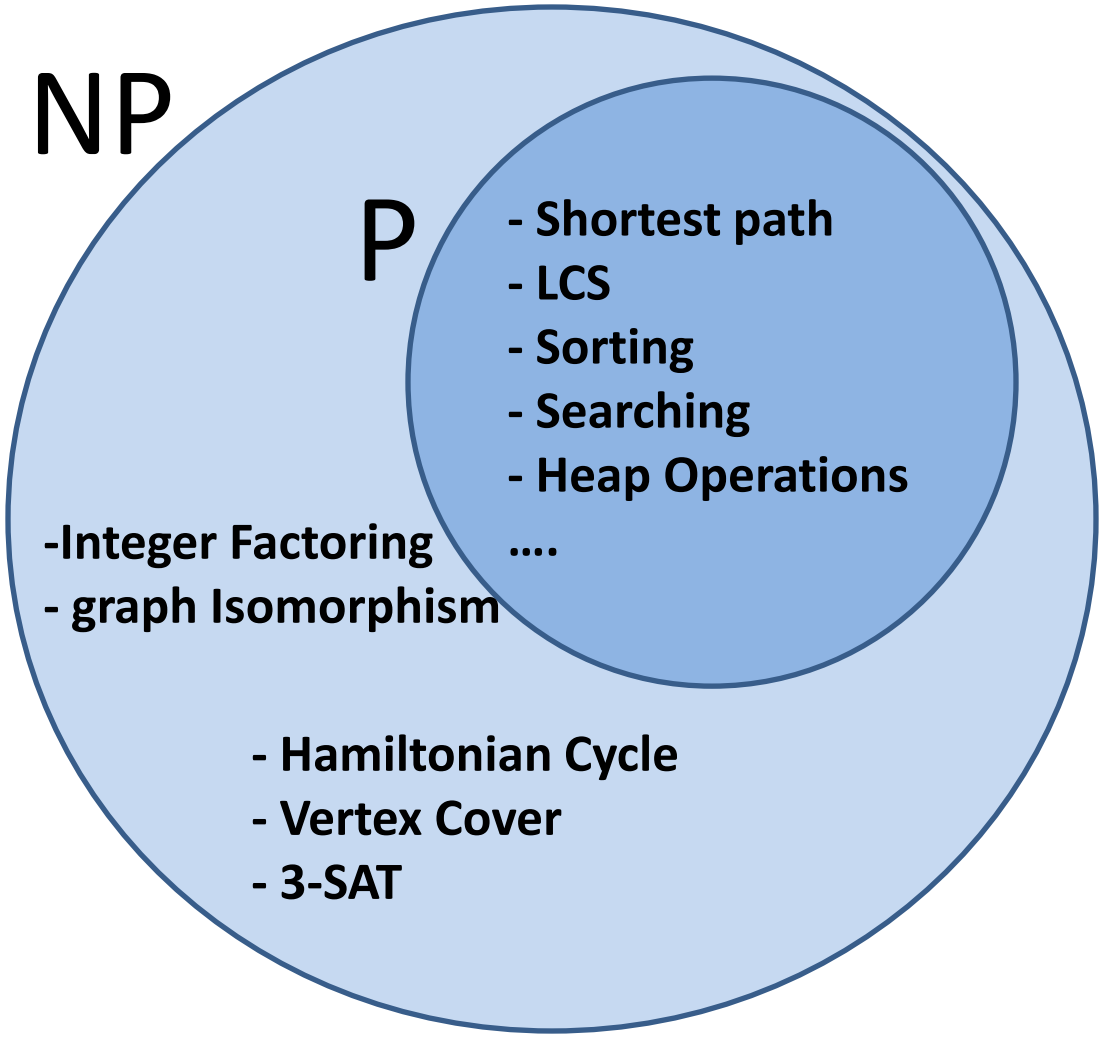
- Following picture shows how exponential number of solutions possible for 3-SAT.
- There are exponential number of path. Each path is a solution.
- A deterministic computer considers one path at one time. That is why, our computers are called deterministic (means, sure. Only one path at a time.)
- A non-deterministic computer can consider all paths at time. That is why it is called no-deterministic (means, not sure of a path.)



Compare P and NP Class

- From previous slide:
  - Non-deterministic computers are more powerful than our computers, because they can solve some problems easily in polynomial time which are hard for our deterministic computers, because our computers cannot solve them in polynomial time.
- Remember: P problems have polynomial solutions in our computers.
- So, P problems are more easy in non-deterministic computers
- In fact, P problems are also NP problems

Compare P and NP



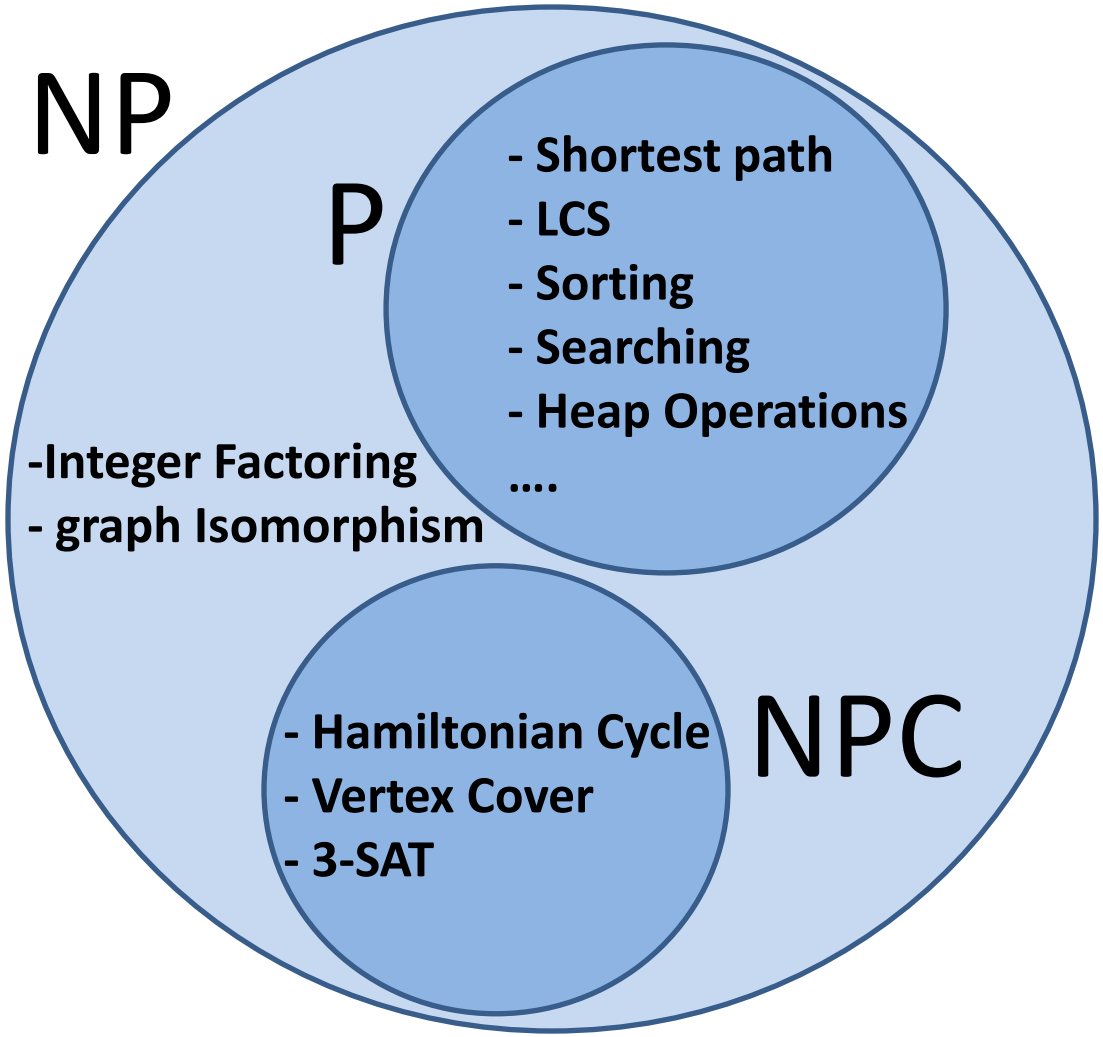
- **P**
  - Shortest path
  - Sorting, searching
  - LCS
  - Heap Operations
  - ……

- **NP**
  - **P**
    - Shortest path
    - Sorting, searching
    - LCS
    - Heap Operations
    - …
  - Hamiltonian Cycle
  - Vertex Cover
  - 3-SAT
  - Integer Factoring
  - Graph Isomorphism
  - …

NP-Complete (NPC) Problems

- What are **NP-Complete** problems?
- Some problems from NP-P are the most difficult, that means hardest for our computer
- All NP problems are easier that those problems or similar hard.
- They are called NP-Complete.
  - Example: Hamiltonian Cycle, Vertex Cover, 3-SAT
- Some problems are NP, but not P or NP-Complete
  - Example: Integer Factorization, Graph Isomorphism
- So, NPC is also a subset of NP
- Also, if we know polynomial algorithm for NPC problems, then all other NP problems can be solved in polynomial time by using those algorithms.

Compare P, NP and NPC



- **P**
  - Shortest path
  - Sorting, searching
  - LCS
  - Heap Operations
  - ……

- **NP**
  - **P**
    - Shortest path
    - Sorting, searching
    - LCS
    - Heap Operations
    - …
  - **NPC**
    - Hamiltonian Cycle
    - Vertex Cover, 3
    - 3-SAT, …
  - Integer Factoring
  - Graph Isomorphism
  - …

How Do You Prove that a Problem is NP-Complete?

- By two steps:
  - (1) Show that the problem is in NP class
  - (2) The problem is same or harder than another known NP-Complete problem. This step is called **reduction.**

How to Know a Problem is in NP Class?

- Take any solution (correct or not)
- Verify whether it is correct or not
- If the time taken in polynomial, then the problem is in NP
- Why? Because, remember, NP means polynomial in non-deterministic machine. So if verifying one solution takes polynomial, then by taking all solutions parallelly by non-deterministic machine, the overall solution time will be polynomial.

How to Know a Problem is in NP Class?

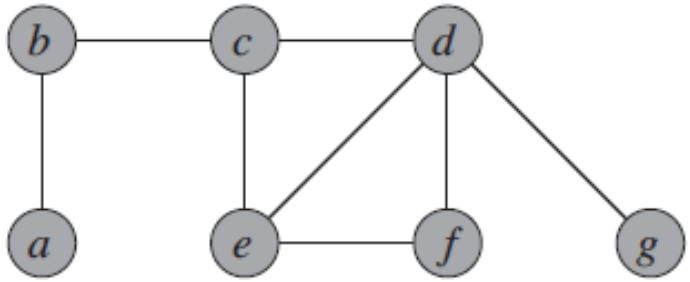
- **Example:** The problem Minimum Vertex Cover is in NP
  - because, if are given a graph and some vertices as the covers, then we can check whether the given vertex set cover all edges or not.
  - Suppose that we are given the following graph and the possible solution
  - Check: c covers (b,c), (c,d), (c,e)

    Time taken: degree of c = 3 steps
  - Check: d covers: (c,d), (d,e), (d,f), (d,g)

    Time take: degree of d = 4 steps
  - Check: e covers: (c,e), (d,e), (e,f)

    Time taken: degree of e = 3 steps
  - So, are all edges covered by c,d,e? NO.

Possible solution: {c,d,e}



  - So, the answer is wrong, that means, {c,d,e} is not a vertex cover.
  - Total time taken: sum of degree of c,d,e = 3+4+3=10 steps.
  - In general for any given answer we can check in total time sum of degree of all vertex <= 2E = $O(n^2)$ time, because E can be at most $O(n^2)$ = polynomial.

How to Know a Problem is in NP Class?

- **Example:** The problem 3-SAT is in NP
  - because, if are given a 3-SAT formula and some 0/1 values of the variables, then we can check whether the values gives final result 1 or 0.
  - Suppose that we are given the following 3-SAT formula and the possible solution
  - Check **$(a \vee b \vee c)$:**

    It is **$(1 \vee 0 \vee 1) = 1$**

    Time taken: 3 steps
  - Check **$(\neg a \vee \neg b \vee \neg c)$:**

    It is **$(\neg 1 \vee \neg 0 \vee \neg 1) = (0 \vee 1 \vee 0) = 1$**

    Time take: 3+3(for negative) = 6 steps
  - Then check **$(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$** :

    It is **$1 \wedge 1 = 1$**
  - So, it gives 1, so YES, the answer is correct.
  - Total time taken: 3 + 6 = 9 steps.
  - In general, for any given answer we can check in total time t most 6*number of clauses = 6*n = O(n) = polynomial.

Possible solution: a=1, b=0, c=1

$(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

How to Know a Problem is in NP Class?

- **Example:** The Problem Clique is in the Class NP.
  - because, if we are given a graph, a set of vertices from that graph, and a value k, then we can check whether every pair of vertices in the set is an edge in constant time. If all pairs are edges and the number of verices in the set is k, then the answer is YES, otherwise NO.
  - Suppose that we are given the following graph and the possible solution (1)

    Check (c,d) edge? Yes.

    Check (c,e) edge? Yes.

    Check (d,e) edge? Yes.

    And, there are 3 vertices.

    So the answer is YES.
  - But if we are given the following possible solution (2), then

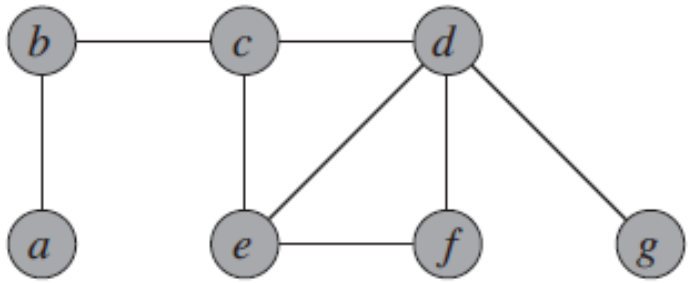
    Check (a,b) edge? Yes.

    Check (b,c) edge? Yes.

    Check (a,c) edge? No.

    So the answer is NO.

Possible solution 1: {c,d,e}, 3



Possible solution 2: {a,b,c}, 3

Reduction

- Steps of Reduction:
  - Take another problem Q that is already known to be NP-complete.
  - Show that P is same or more difficult than Q. How?
  - Show that for every example e of Q there is an example e' in P such that the **Solution of e $\Leftrightarrow$ Solution of e'.**

Remember: $\Leftrightarrow$ means $\Rightarrow$ and $\Leftarrow$

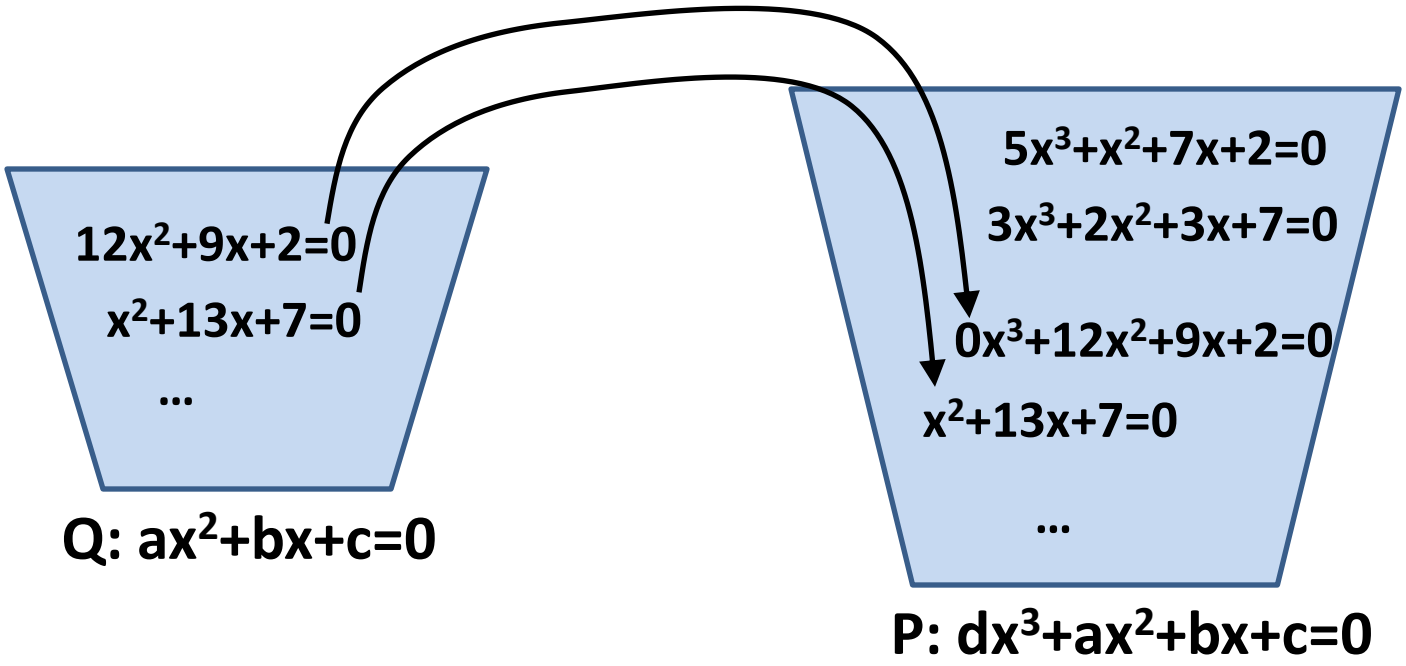
- If we can do the reduction, then we can say that **Q is reduced to P**.

Example of Reduction

- In this example, we show that the problem P: **Solve $dx^3+ax^2+bx+c=0$** is same or more difficult than Q: **Solve $ax^2+bx+c=0$** by reduction.
- Take any example of Q, **e': $12x^2+9x+2=0$**
- The example of P that is same as e' is e: dx^3+ax^2+bx+c with d=0, a=12, b=9, c=2. So, e is: **$0x^3+12x^2+9x+2=0$**, that means, e': **$12x^2+9x+2=0$.** So, e and e' are same and their solutions are also same.

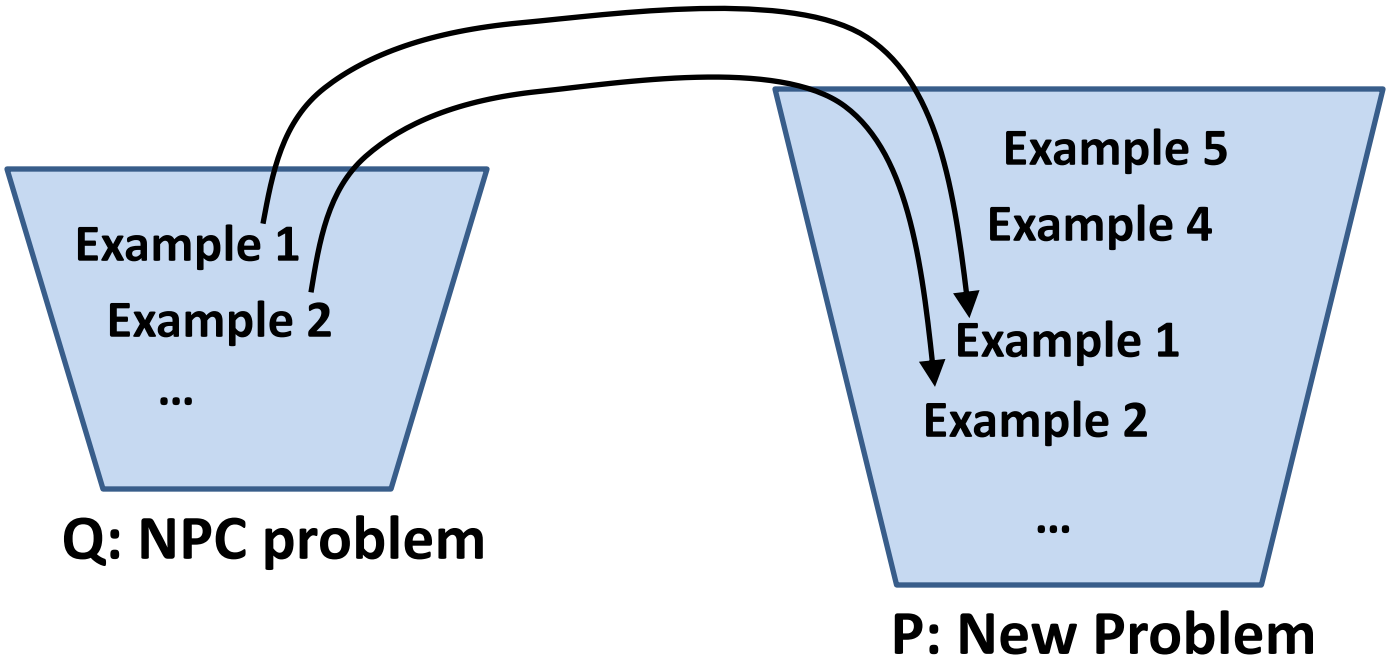
What Do We Know From Reduction?

- If we can reduce problem Q to problem P, then we can say that P is same or harder than Q.
- Because, every example e' of Q can be solved by the corresponding example e' of P.



What Do We Get From Reduction?

- If we can reduce problem Q to problem P, then we can say that P is same or harder than Q.
- So, if Q is NP-Complete, then P is same or harder than Q. That means, P is also NP-Complete.



Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- **Two steps:**
  1. For every example e of 3-SAT, make an example e' of Clique
  2. Prove that solution of e $\Leftrightarrow$ solution of e'. By,
     - 2a. Show that solution of e $\Rightarrow$ solution of e'
     - 2b. Show that solution of e' $\Rightarrow$ solution of e

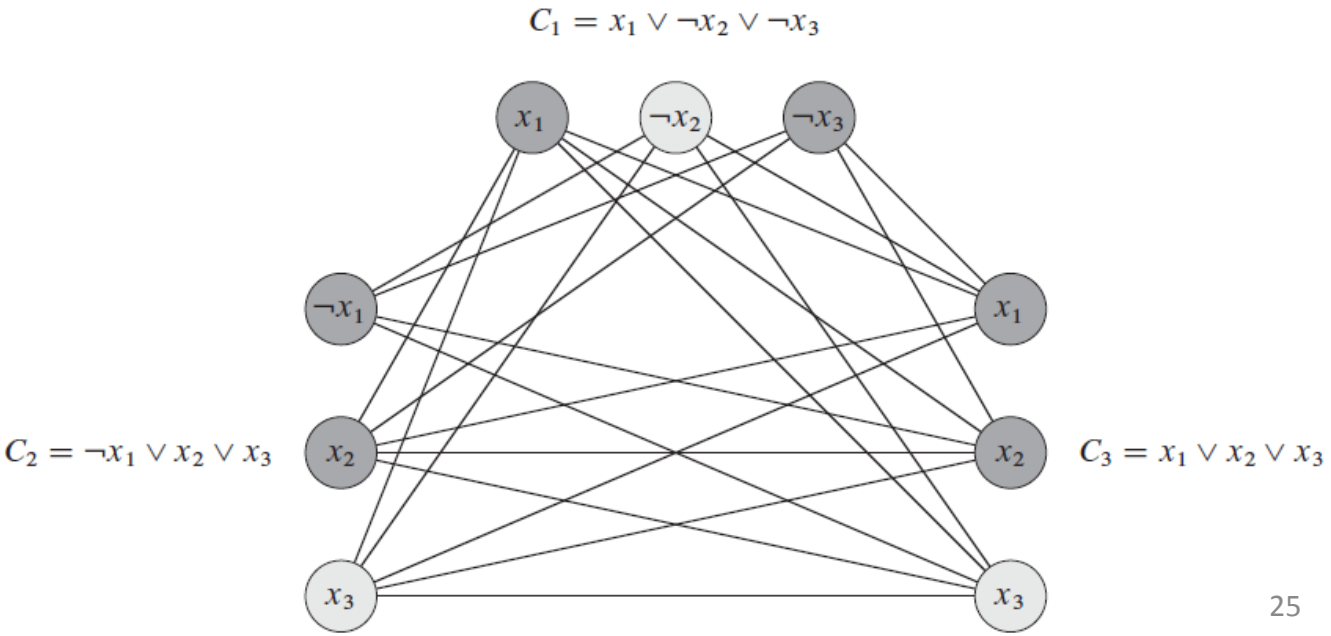
- **Step 1:** For every example e of 3-SAT, make an example e' of Clique
- Example of 3-SAT, e: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$
- Example of Clique, e': a graph, and clique size k.
- So, we need to make a graph and find a k from a 3-SAT expression.

Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- **Step 1:** For every example e of 3-SAT, make an example e' of Clique. That mens a graph and a clique size k.
- For every literal in e, take one node in e'. Connect two nodes in e' if they are not negative of each other or they are not in the same clause. Take clique size = number of clauses.
- Example: Suppose e: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

  Then the graph is the picture below, and the clique size = 3.

e'

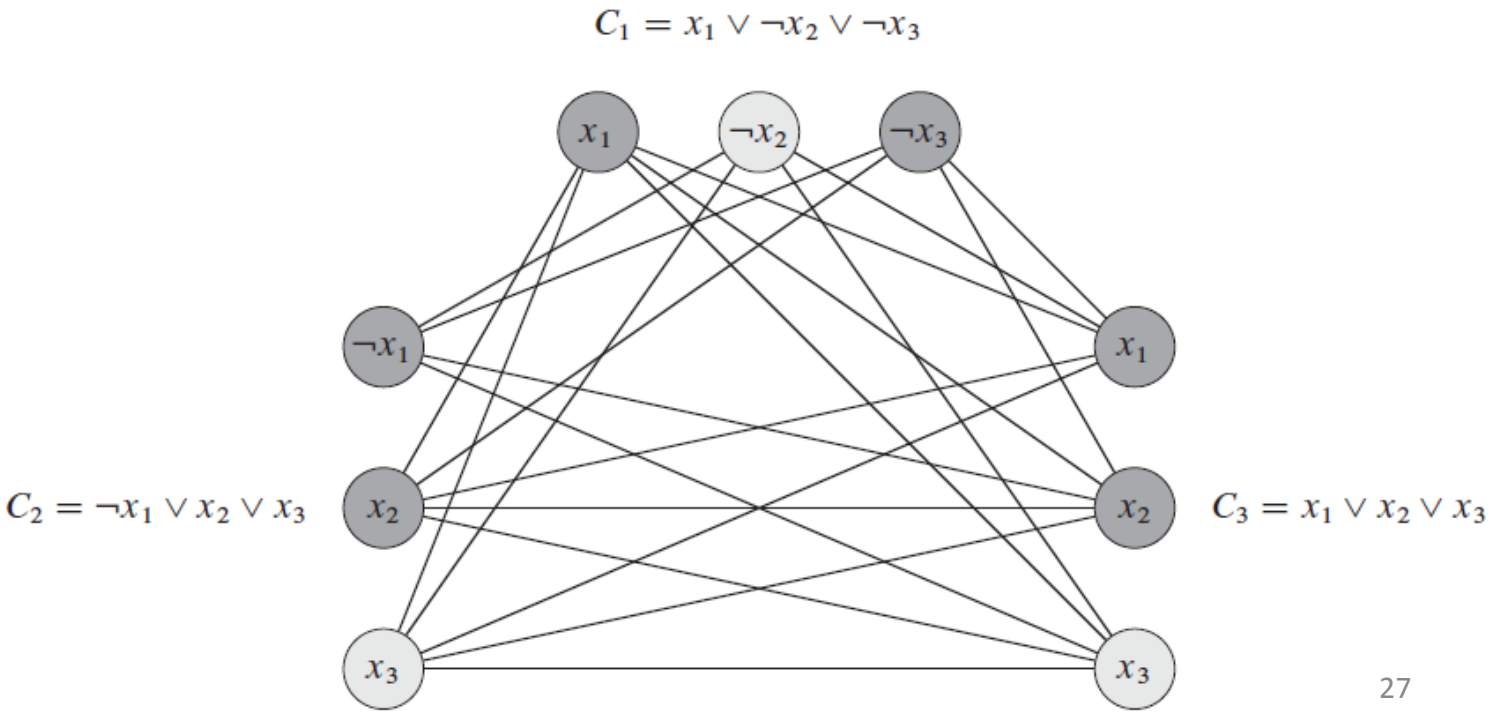


Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- **Step 2:** Prove that solution of e $\Leftrightarrow$ solution of e'.
- **Step 2a:** Prove that, solution of e $\Rightarrow$ solution of e'. That means, if we know solution of e, then we can find solution of e'.
  - **Proof:** If there is a solution of e, then each clause is 1. For example,
    $(x_1 \vee \neg x_2 \vee \neg x_3)$ = 1
    $(\neg x_1 \vee x_2 \vee x_3)$ = 1
    $(x_1 \vee x_2 \vee x_3)$ = 1
  - So, each clause has at least one literal whose value is 1. Take the corresponding vertex in e' in clique.
  - Because the literals with value 1 is not negative of each other, so they have edge in e'. That means they form a clique.
  - Total size of the clique is k = number of clauses.

Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- For example, in e, a satisfying assignment is x2=0, x3=0 and x1 = 0 or 1. So, e=1.
- Now, in e', take ¬x2 from Clause 1, x3 from Clause 2 and x3 from Clause 3. Then they form a clique of size 3.
- So, if we know a solution for e, then we can find a solution for e'.

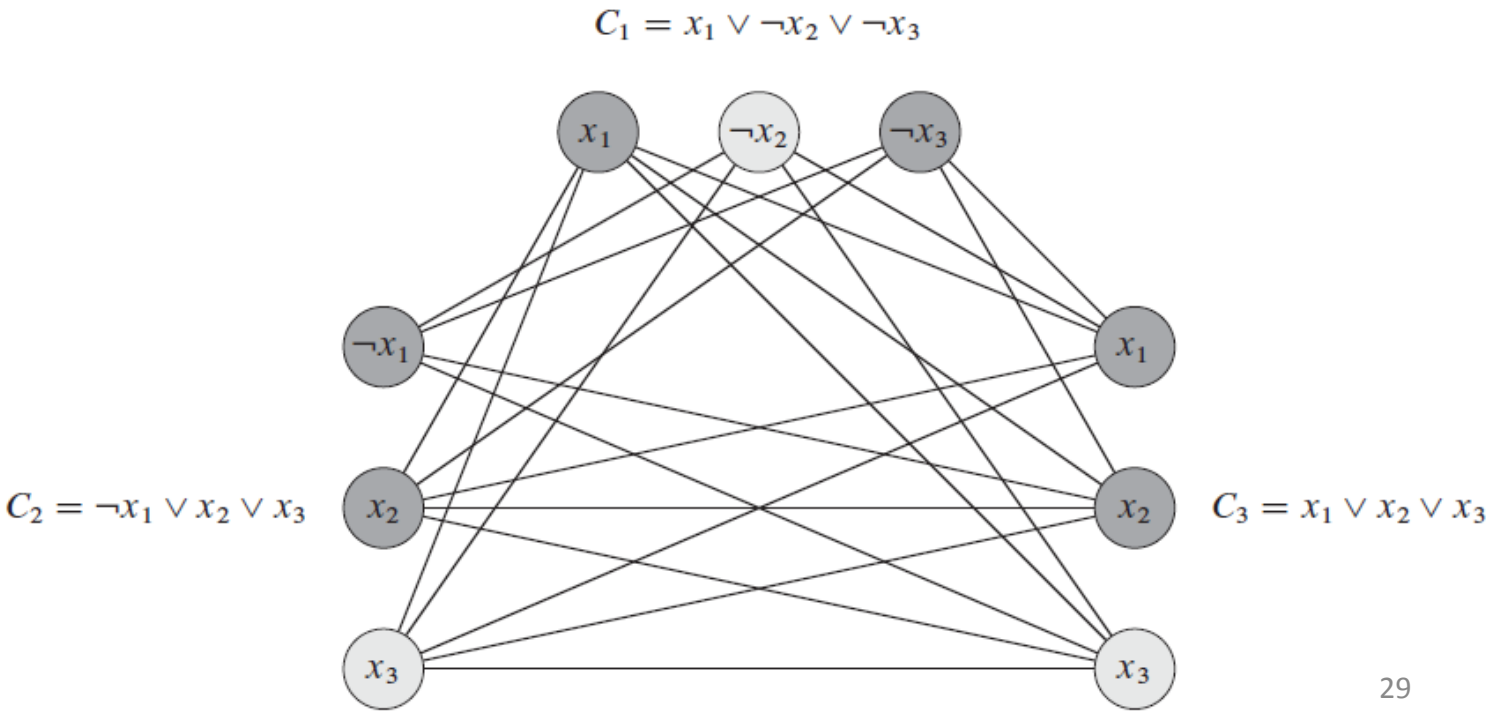


Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- **Step 2b:** Prove that, solution of e' $\Rightarrow$ solution of e. That means, if we know solution of e', then we can find solution of e.
  - **Proof:** If there is a solution of e', then there is a clique of size k.
  - Because there is no edge between two literals in same clause, the clique contains exactly one vertex from one clause.
  - Make that literal as 1. Because there is no edge between 0 and 1 in two clauses, it will not assign two negative literals as 1.

Example of NP-Complete Reductions
Reduce 3-SAT to Clique

- **Step 2b:** For example, take x1 from Clause 1, x2 from Clause 2 and x3 from Clause 3. They form a clique of size 3.
- Now, in e, make x1=1, x2=1 and x3=1. This is a solution for e, because it makes all three clauses 1.
- So, if we know a solution for e', then we can find a solution for e.



How To Prove Clique in NP-Complete?

- **Step 1:** From Slide 19, we can prove that Clique is in Class NP.
- **Step 2:** From Slides 24-30 we know that 3-SAT can be reduced to Clique. So, Clique is same or more difficult than 3-SAT. Since 3-SAT is already known to be NP-Complete, so Clique is also NP-Complete.