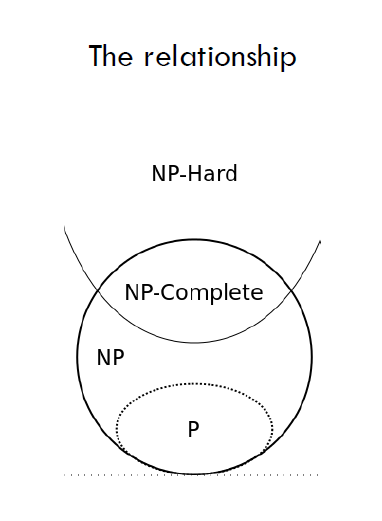
**Lecture 10 & 11 Hardness Summary**

**The relationship between P and NP and NP-Complete:**

**Three ways of thinking of reductions.**

Reduction by simple equivalence

Reduction from special case to general case

Reduction by encoding with gadgets

**Polynomial-Time Reduction:**

Problem Y can be solved in polynomial time.

Problem X polynomial reduces to problem Y, denoted as X<=pY, if arbitrary instances of problem X can be solved using

1. polynomial number of standard computational steps, and
2. polynomial number of calls to an oracle that solves problem Y.

E.g. Transitive Closure <= p BFS

Transitive closure of a directed graph can be computed by n calls to BFS plus the time to build the transitive closure.

**Steps of Reduction:**

We have input to X, we transform instance for X to instance for Y, then we solve Y, and we get solutions to Y of input f(x). Then we transform solutions for Y to a solution for X. Done.

我们先解决Y，再用Y的解来解决X。

**Polynomial Time Reduction**:

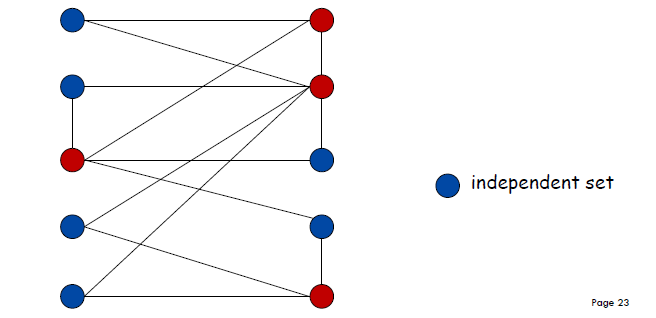
If X<=pY and Y can be solved in polynomial-time, then X can also be solved in polynomial time.

If X<=p Y and X cannot be solved in polynomial time, then Y cannot be solved in polynomial time.

Self-reducibility: Search Problem <=p Decision Problem

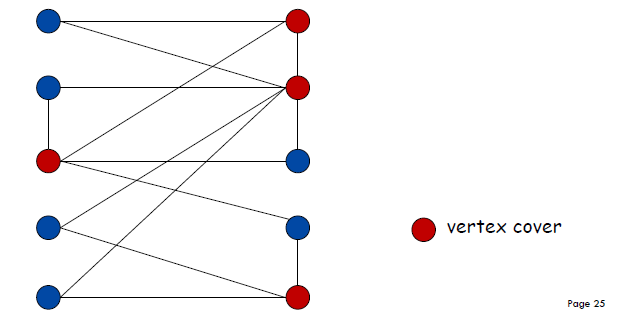
**Reduction by Simple Equivalence**

**Independent Set**: Given a graph G=(V , E) and an integer k, is there a subset of vertices S属于V such that |S|>=k, and for each edge AT MOST ONE of its endpoints is in S.



**Vertex Cover**: Given a graph G=(V , E) and an integer k, is there a subset of vertices S属于V such that |S|<=k, and for each edge, AT LEAST ONE of its endpoints is in S.

Vertex-Cover =p Independent-Set (This is a reduction by simple equivalence.)

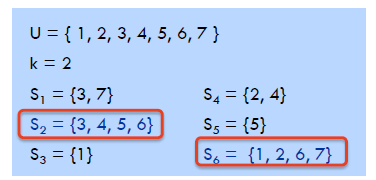


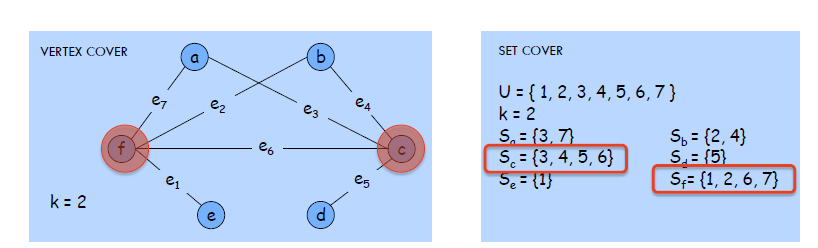
INDEPENDENT-SET =P VERTEX-COVER

**Set-Cover**: Given a set U of elements, a collection of subsets of U, and an integer k, does there exist a collection of k of these sets whose union is equal to U.

Vertex-Cover can be reduced to Set Cover Problem. (Reduction from special case to general case.).

mathematically, Vertex-Cover <=p Set-Cover





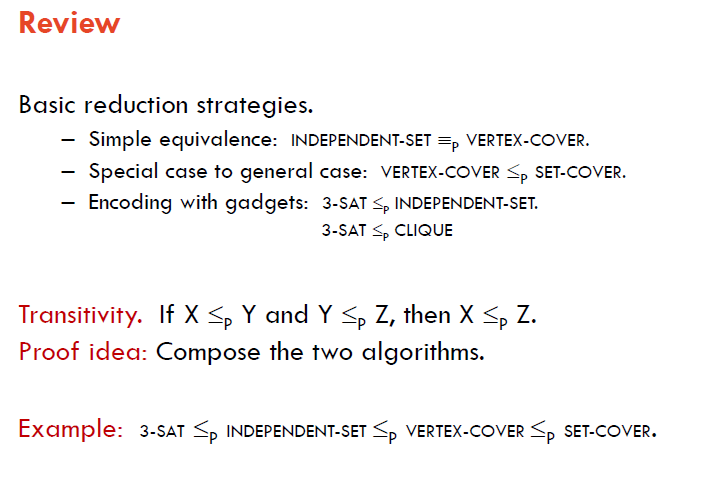
3-Satisfiability problem reduces to Independent Set problem.

3-SAT <=P INDEPENDENT-SET

Clique: A clique of a graph G is a complete subgraph of G. A graph can have multiple cliques.

PS: Complete Graph: A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

3-SAT <=p CLIQUE



**Class P**: Decision problems for which there is a polynomial time algorithm.

**Class NP**: Decision problems for which there exists a polynomial time certifier.

Certifier: It doesn’t solve the problem by its own, rather, it checks if a certificate is a valid solution.

SET-COVER is in NP

SAT is in NP

**Hamiltonian Cycle**: Given an undirected graph G = (V , E), does there exist a **simple cycle** C that **visits every node**.

**Directed Hamiltonian Cycle**: Given a **directed graph** G = (V , E), does there exists a simple directed cycle that contains every node in V.

Ham-Cycle is in NP;

Dir-Ham-Cycle is in NP;

Theorem: DIR-HAM-CYCLE<=p HAM-CYCLE

Theorem: 3-SAT<=p DIR-HAM-CYCLE

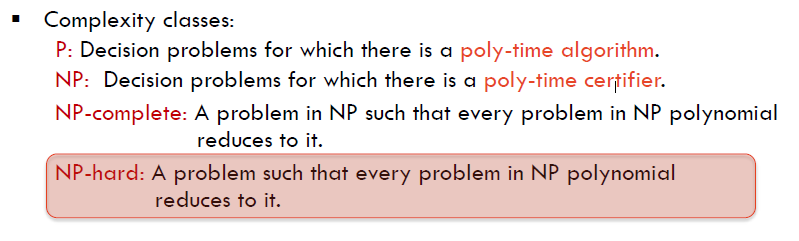
**Longest Path problem**: Given a digraph G = (V, E), does there exists a simple path of length at least k edges?

Theorem: 3-SAT <=p LONGEST-PATH

**Class P belongs to Class NP**

**Class NP-Complete**: A problem in NP with the property that for every problem in NP polynomially reduces to it.

**Class NP-Hard:** A decision problem such that every problem in NP polynomial reduces to it. (Not necessarily in NP)



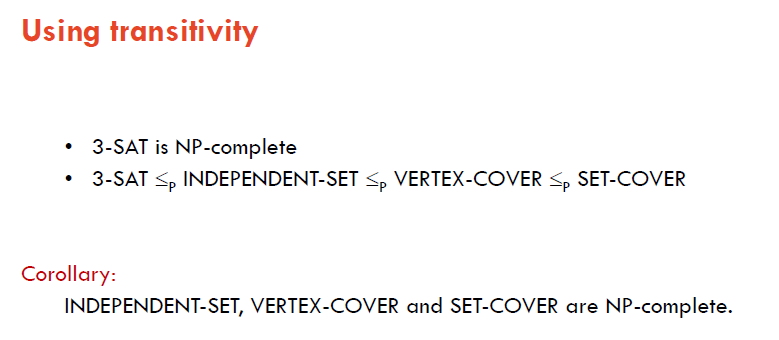
CIRCUIT-SAT: Given a combinational circuit built out of AND, OR and NOT gates, is there a way to set the circuit inputs so that the output is 1.

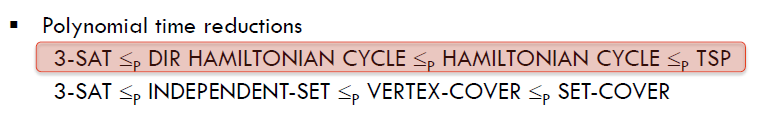
**CIRCUIT-SAT is NP Complete**.

Justification: If X is an NP-Complete problem, and Y is a problem in NP with the property that X<=pY then Y is NP-Complete, which means if Y is in NP, and there exist a NP-Complete problem X which can be reduced to Y in polynomial time, then Y is NP-Hard, therefore, it is NP-Complete.

3-SAT is NP-COMPLETE

Transitivity:





Travelling Salesman Problem

Given a set of n cities and a pairwise distance function d(u , v), is there a tour of length <= D

Theorem: HAM-CYCLE <=p TSP

HAM-CYCLE is NP-Complete

TSP (decision version) is NP -Complete

TSP is in NP

**Solving Restricted Instances: Finding small vertex cover**:

**Theorem**: Let (u , v) be an edge in graph G, then G has a vertex cover of size <= k if and only if at least one of G\{u} and G\{v} has a vertex cover of size <=k-1.

**Theorem:** Vertex cover can be solved in O(2k kn) time.

Observation: If G has a vertex cover of size k, it has <= k(n-1) edges.

**Independent Set on Trees**

Given a graph G = (V , E) and an integer k, is there a subset of vertices S 属于V such that |S|>=k, and for each edge at most one of its endpoints is in S? And given a tree, we need to find this maximum Independent Set.

Theorem: INDEPENDENT-SET on trees can be solved in O(n) time.

**Weighted Independent Set on Trees (Assignment 3)**

This can be solved using Dynamic Programming.

And DP find the solution in O(n) time.

**Approximation Algorithm**:

Not optimal solution, but within a guaranteed error.

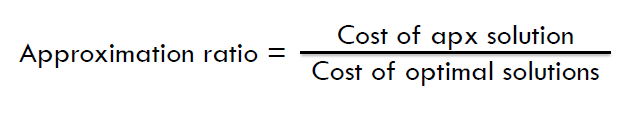
**Approximation Algorithm: Load Balancing**

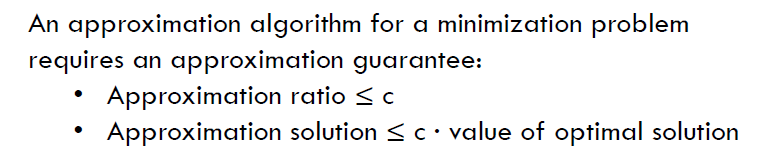
**Load Balancing: List Scheduling**

Makespan: it is the maximum load on any machine L.

Load balancing: assign each job to a machine to minimize makespan.

**Approximation Ratio**





Theorem: Greedy Algorithm is a 2-approximation algorithm.

**Prove a problem is in P:** By finding a polynomial time algorithm that solve this problem.

**Prove a problem is in NP:** Firstly, find a certificate. Secondly, find a certifier. Thirdly, show that this certifier can verify whether the certificate is TRUE or not. Fourthly, prove that the certifier runs in polynomial time by analysing the running time line by line and add them together.

**Prove a problem L is in NP-hard:** Firstly, select a known NP-complete problem L’. Secondly, describe an algorithm f that transforms L’ into L. Thirdly, prove that the algorithm f is correct. Fourthly, prove that f runs in polynomial time.

**Prove a problem is in NP-complete**: Firstly, prove that this problem is in NP. Secondly, prove that this problem is in NP-hard.

**Recipe to establish NP-Completeness of problem Y**:

1. Show that Y is in NP
2. Choose a known NP-Complete problem X.
3. Prove that x<=p Y, which means Y is in NP-Hard

Done~

**Justification**: If X is an NP-Complete problem, and Y is a problem in NP with the propertythat X<=p Y then Y is NP-Complete.