# Week 13 - 02
# P & NP contd.

50.004 Introduction to Algorithm

Dr. Subhajit Datta
Based on original slides by Dr. Simon LUI
ISTD, SUTD

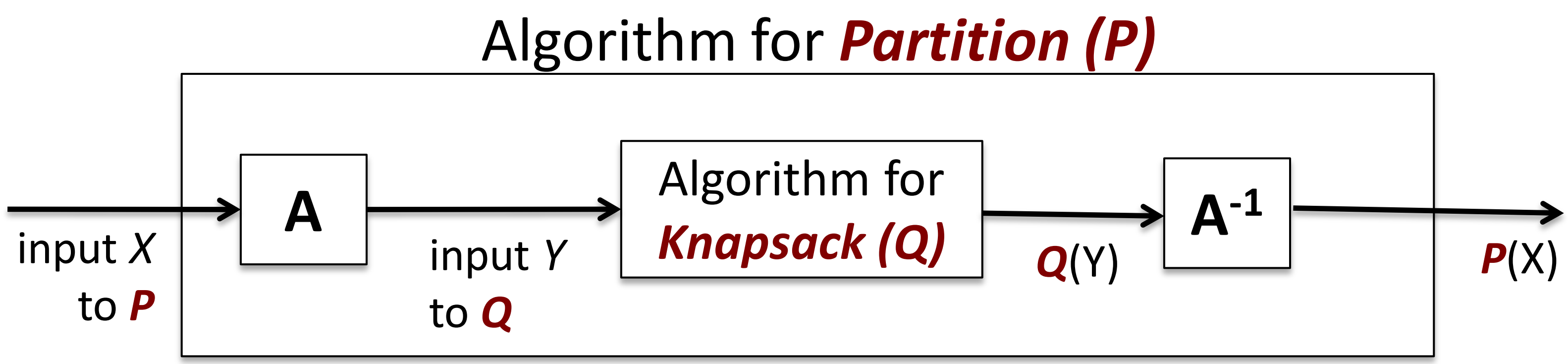
# POLY REDUCTION - EXAMPLE

# Example: Knapsack

- A “believed hard” problem is **Partition**:
  - Given a set of $n$ numbers summing to $S$.
  - Is there a subset of numbers summing to $S/2$?
- We can use this to show **Knapsack** is hard
  - Suppose we have an algorithm A for **Knapsack.**
  - Want to use it to solve **Partition**. How?

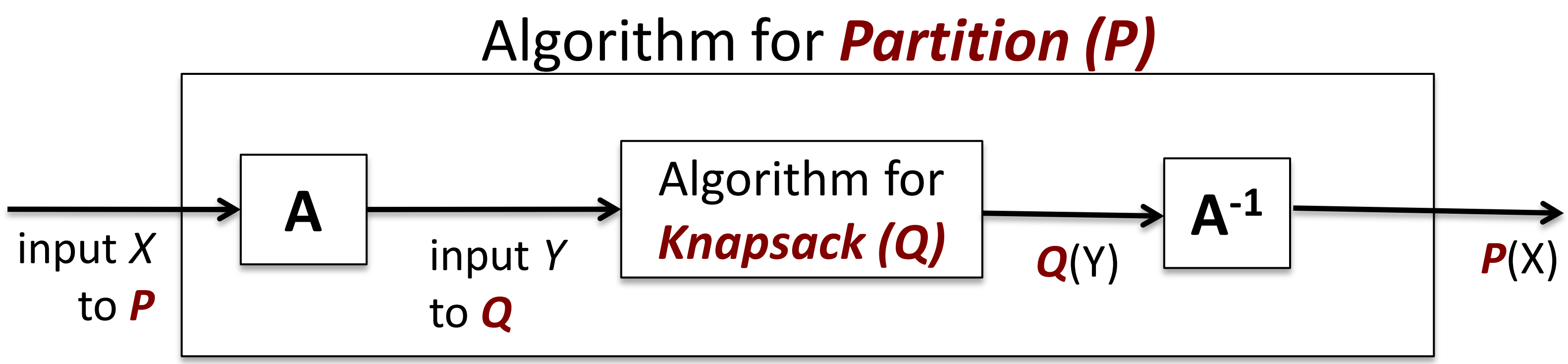
# Example: Knapsack

- A "believed hard" problem is **Partition**:
  - Given a set of *n* numbers summing to *S.*
  - Is there a subset of numbers summing to *S*/2?
- We can use this to show **Knapsack** is hard
  - Suppose we have an algorithm A for **Knapsack.**
  - Want to use it to solve **Partition**. How?



# Example: Knapsack

- Given an input $\{s_1,\ldots,s_n\}$ to **Partition.**
  - Consider the **Knapsack** problem where item $i$ has size $s_i$ and value $v_i$, and knapsack size is $S/2$.
  - If there is a partition, you can fill the knapsack and get value = $S/2$.
  - Otherwise, best achievable value is < S/2.



# Example: Knapsack

- We now have an algorithm for **Partition**:
  - Convert the **Partition** input to **Knapsack** input, in poly time
  - Run **Knapsack**.
  - Convert the **Knapsack** answer into a **Partition** answer, in poly time
  - If Knapsack result is S/2, return YES, else return NO

# Summary so far

- If problem $\boldsymbol{P}$ is reduced to problem $\boldsymbol{Q}$, $\boldsymbol{P} \leq_P \boldsymbol{Q}$
  - this shows $\boldsymbol{Q}$ is at least as hard as $\boldsymbol{P}$.
- If people think $\boldsymbol{P}$ is hard, they'll believe $\boldsymbol{Q}$ is hard.
- Problem: what is a hard $\boldsymbol{P}$ to use for $\boldsymbol{P} \leq_P \boldsymbol{Q}$ ?
  - Is there a problem that everyone agrees is hard despite not being able to prove it?
- Solution: Find a whole family of hard problems that can be simultaneously reduced to $\boldsymbol{Q}$.

$$\boldsymbol{P_1} \cong_P \boldsymbol{P_2} \cong_P \ldots \cong_P \boldsymbol{P_n} \leq_P \boldsymbol{Q}$$

# NP PROBLEM
# NPC PROBLEM
# REDUCTION OF NPC TO NPC

# *NP*

- A decision problem belongs to the class **NP** if:
  - it always has a poly-size solution;
  - Given a correct solution, it can be **verified** in poly-time.
- We say that such problem can be solved in **nondeterministic polynomial time (NP)**.
  - We can **guess** the solution
  - then in poly-time **check** whether our guess is right.
- E.g. Is there a path of length greater than L?
  - We can guess a path, then check if its length is larger than L.

# The hardest problems in NP

- A problem ***Q*** is **NP-hard** if
  - all NP program can be reduced to it: for all ***Z*** in *NP*, ***Z*** $\leq_P$ ***Q***
  - ***Q*** can be turned into any other NP problem, in poly time
  - ***Q*** is at least as hard as any NP problem
- A problem ***Q*** is **NP-complete (NPC)** if it is in NP and NP-hard
  - ***Q*** Is the hardest problem in NP
  - ***Q*** is in NP, and for all ***P*** in *NP*, ***P*** $\leq_P$ ***Q***
- SAT is an NP-complete problem! (Cook, 1971)
- It is a good starting point for showing other problems are hard.

# Example of a reduction

- The 3-SAT problem is NP-complete
- The K-Graph Independent Set (K-GIS) problem is in NP but we don't know if it is hard
- Now, let's reduce the 3-SAT to K-GIS using a poly-reduction.

# The 3-SAT problem

- **SAT** (Satisfiability): given a boolean formula, can you make it TRUE;

$$(x_1 \wedge (x_2 \vee \bar{x}_3)) \wedge ((\bar{x}_2 \wedge \bar{x}_3) \vee \bar{x}_1) \Rightarrow x_1 = 1, x_2 = 0, x_3 = 0$$

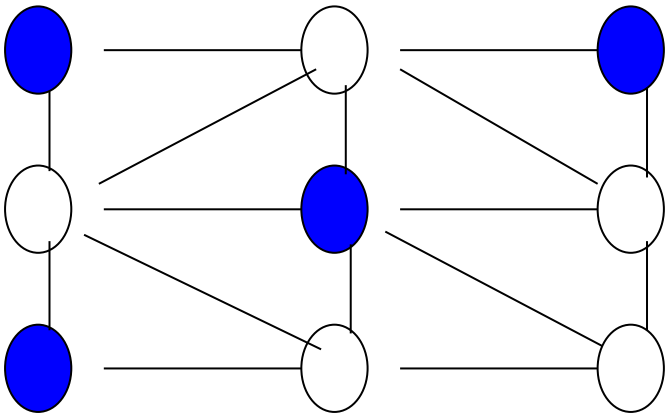
- **3-SAT**: AND clauses, each clause contains 3 variables by OR. For example:

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

- **Cook's Theorem**: 3-SAT is NP-complete

# K-Graph Independent Set (K-IS)

- Set of K nodes, all pairs are NOT adjacent to each other
- For example, the following blue nodes are 4-IS (k=4)

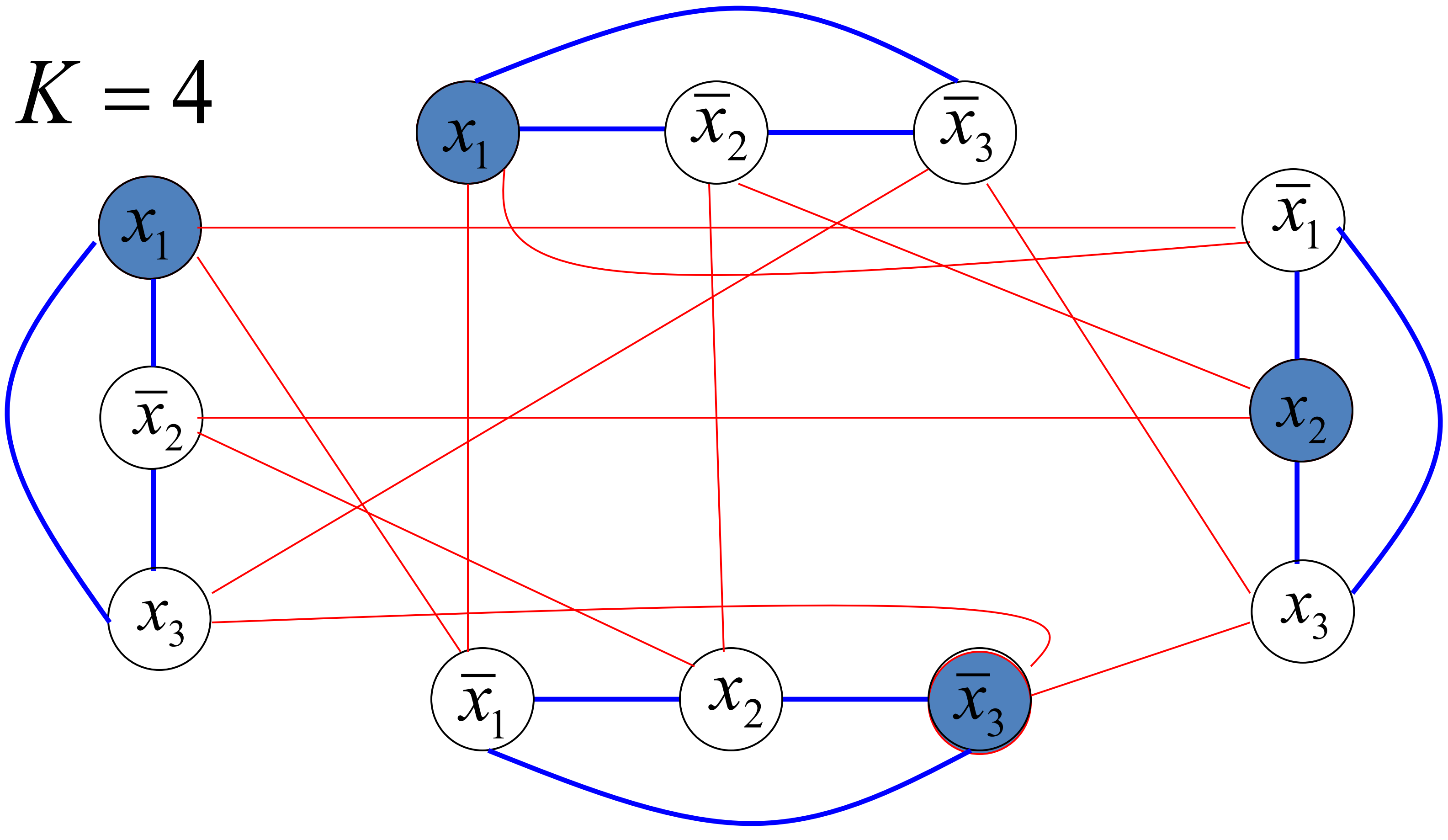


- QS: Is K-IS NP-complete?

# Reduction of 3-SAT to K-IS

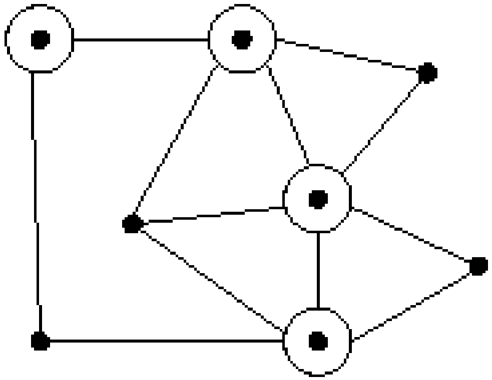
$$(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3)$$

$$K = 4$$



# Exercise 1

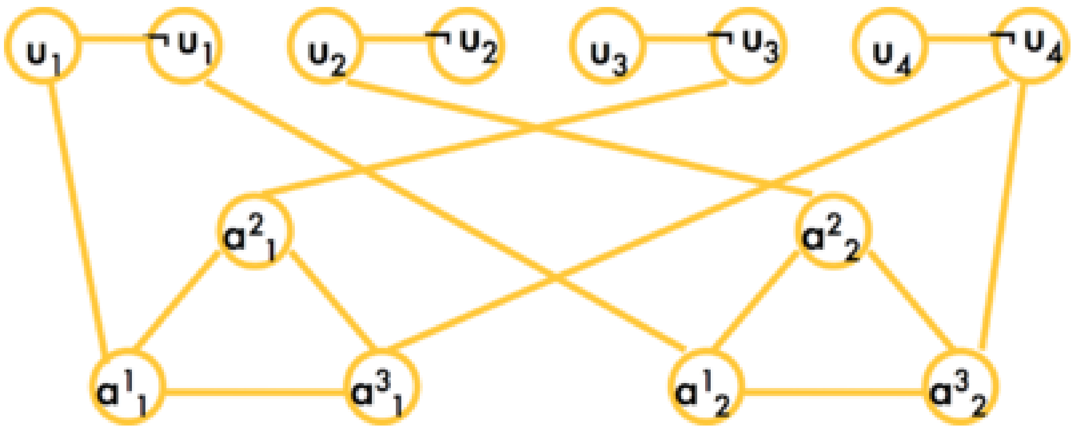
- Vertex Cover (VC): is there a subset of at most k vertices, such that it connect to all edges?



e.g. in this graph, 4 of the 8 vertices is enough to cover

- Exercise: Proof that VC is NP Complete
  - Solution:
    - first, proof that it is in NP (easy)
    - then Reduce 3-SAT to VC(next slide)

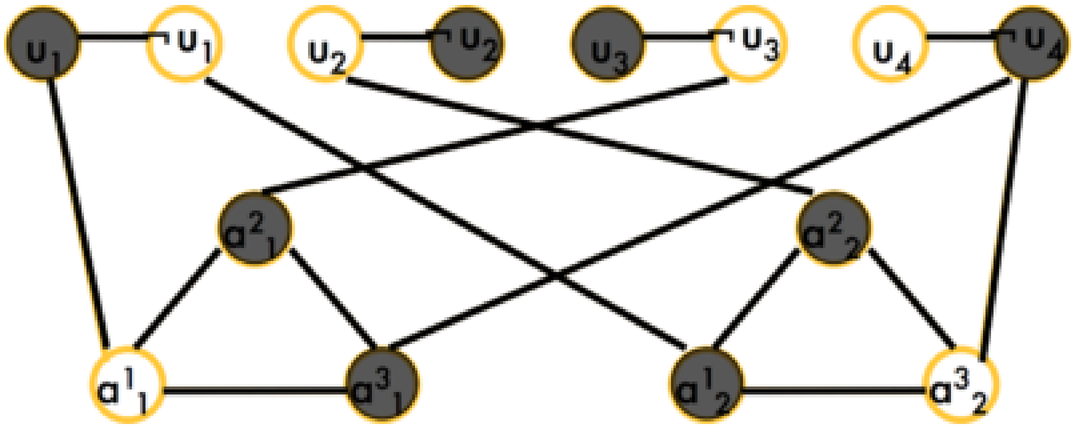
# Solution 1




$$(u_1 \vee \neg u_3 \vee \neg u_4) \wedge (\neg u_1 \vee u_2 \vee \neg u_4)$$

- Setting:
  - v1-v1', v2-v2', v3-v3', v4-v4'
  - 3-SAT component below (as triangle), connected to the same node above
- To cover all edges, we need n vertices, one for each top pair.

# Solution 1

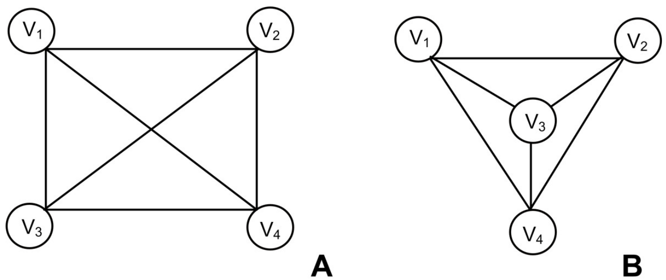



$$(u_1 \vee \neg u_3 \vee \neg u_4) \wedge (\neg u_1 \vee u_2 \vee \neg u_4)$$

- IF 3-SAT can be solved (E.g. u1 u2 u3 u4 = T F T F)
  - Color the true node at the top
  - Color the other nodes at the bottom

# Exercise 2

- K-clique: k vertices, all vertices are adjacent to each other
  - E.g. both of these are 4-CLIQUE

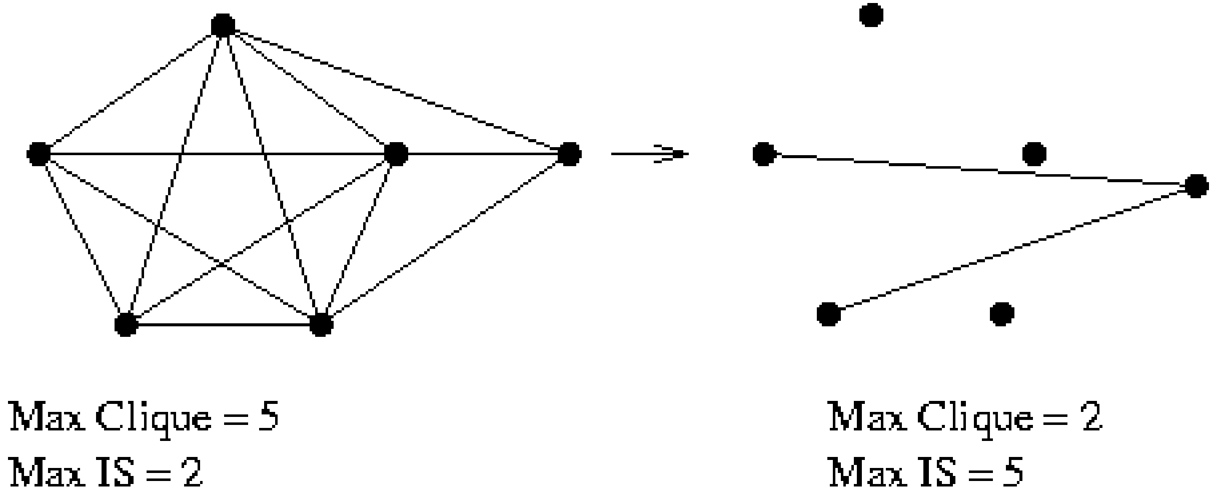



- CLIQUE Problem: in a graph, does k-clique exists?

- Exercise: Proof that CLIQUE is NP Complete
  - Solution:
    - First, proof that it is in NP (easy)
    - Then, reduce Independent set to CLIQUE

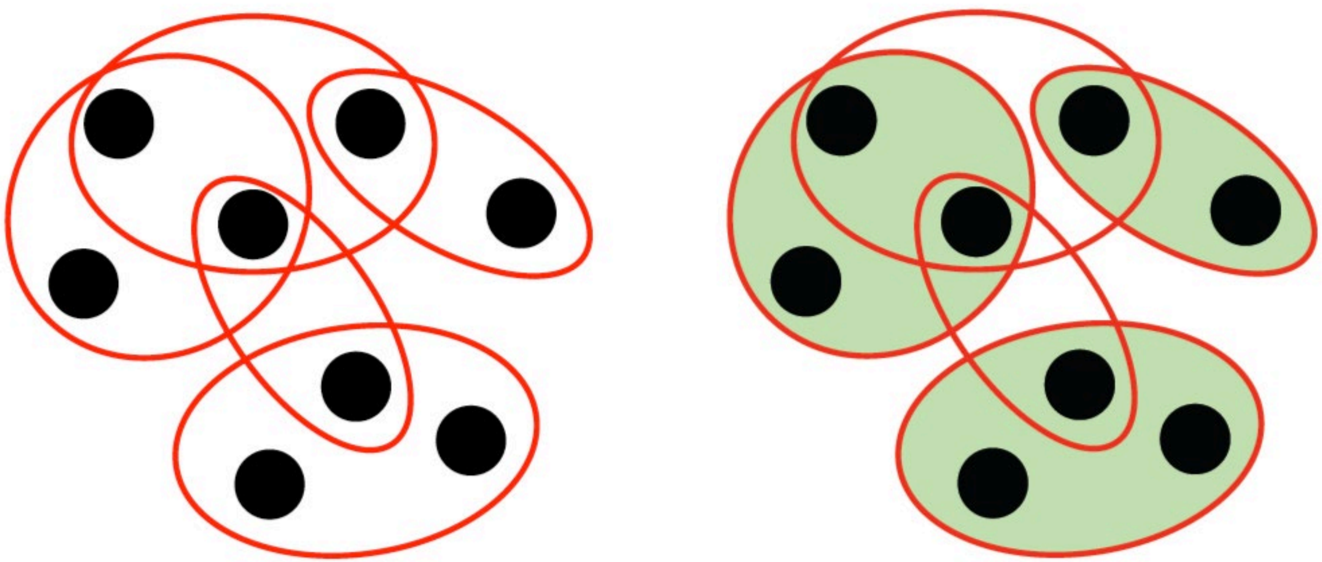
# Solution 2

- Reduce Independent set (IS) to CLIQUE
  - Complement a graph
  - CLIQUE become IS, IS become CLIQUE
  - (most reduction are complicated, this is exceptionally simple…)



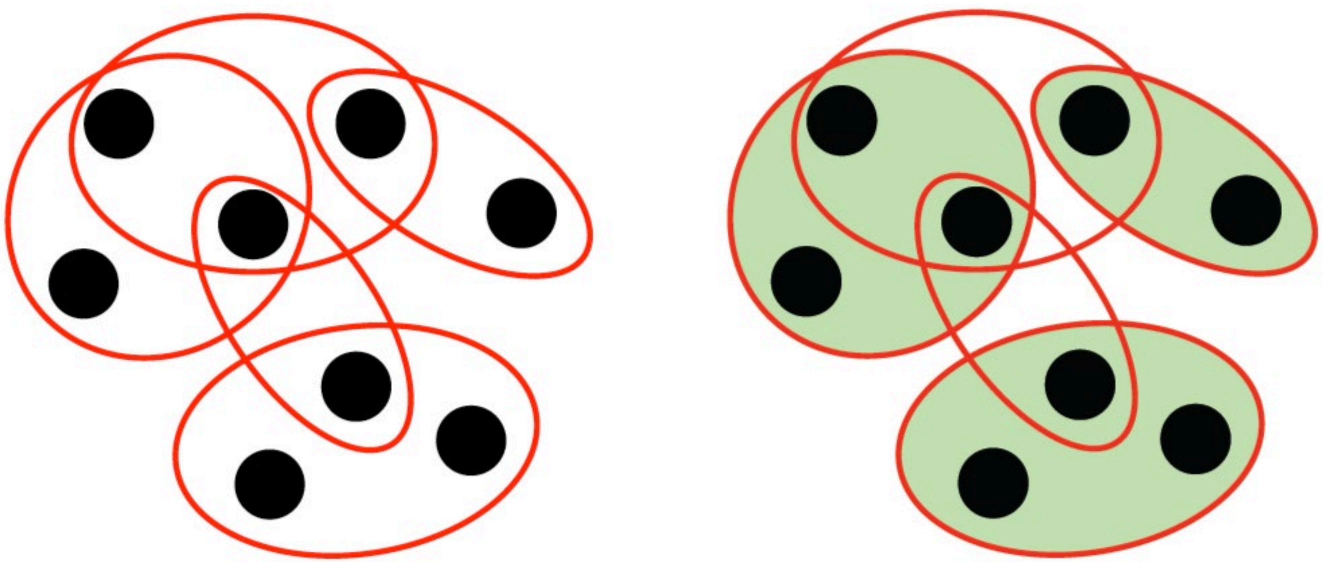
# Exercise 3

- Set Cover: Given a set of U of elements and collection of set $S_1$ $S_2$ $S_3$… $S_m$ of subset of U. Is there a collection of at most k set, whose Union is U?



# Exercise 3

- PROBLEM: Prove that Set Cover is NPC
  - Solution:
    - First, prove that it is NP (Easy)
    - Then, prove that vertex cover can reduce to set cover (next slide)



# Solution 3

- Let G = (V,E) and k be an instance of vertex cover
- Now,
  - U = E (set of edges)
  - Create set of s1, s2, s3….
    - S1 = all edges adjacent to node 1
    - S2 = all edges adjacent to node 2
    - Etc
- Conclusion: If G has a vertex cover of size <=k, then U has a set cover <=k

# P vs NP

- Many problems have been proven to be NP-complete
  - Clique, Independent Set, TSP, Graph Coloring, 4-way matching, Vertex Cover, Hamiltonian Path, Longest path, Multiprocessor Scheduling, Max-Cut, Constraint Satisfaction, Quadratic Programming, Integer Linear Programming, Disjoint Paths, Subset Sum…
  - So not just one, but many "hardest problems in NP"
- In 50+ years, scientists haven't found a polynomial-time algorithm for any of them.
- (A poly-time algorithm for one of them, implies a poly-time algorithm for all, as all are reducible to each other)
- The "P vs NP" problem, i.e. answering whether or not there is a poly-time algorithm for any of these problems, is one of the seven millennium prize problems.
- The Clay Mathematics Institute offers $1million for its answer.