

NP-Complete Problems

P and NP

Polynomial time reductions

Satisfiability Problem

Clique Problem

Vertex Cover

Dominating Set

Chapter 3

Algorithm Design
Kleinberg and Tardos

Polynomial Algorithms

- Problems encountered so far are polynomial time algorithms
- The worst-case running time of most of these algorithms is $O(n^k)$ time, for some constant k .
- All problems cannot be solved in polynomial time
- There are problems that cannot be solved at all – **Unsolvable**
- There are problems that can be solved but not in $O(n^k)$ time for some constant time.
- Problems that are solvable in polynomial time by polynomial-time algorithms are said to be tractable (or **easy or efficient**).
- Problems that require superpolynomial time are said to be **intractable or hard**.

P and NP

- Class P problems are solvable in polynomial time.
- Class NP problems are verifiable in polynomial time.
 - For example, given a problem, we can verify the solution in polynomial time
- Any problem in P is also in NP
- $P \subseteq NP$
- It is **NOT KNOWN** whether P is a proper subset of NP.

NP

- **What are NP-complete Problems?**
A problem is said to be NP-Complete if it is as 'hard' as any problem in NP.
- No polynomial time algorithm has yet been discovered for an NP-Complete problem.
- However, it has not been proven that NO polynomial time algorithm can exist for an NP-Complete problem.
- This problem was first posed by Cook in 1971.
- The issue of, $P=NP$ or $P \neq NP$ is an open research problem

Examples of NP-Complete problems

- Shortest vs. longest simple paths
- Finding the shortest paths from a single source in a directed graph $G=(V,E)$ can be completed in $O(V E)$ time. Even with negative edge weights.

However, finding the longest simple path between two vertices is NP-complete. It is NP-Complete even if each of edge weights is equal to one.

- **An Euler tour of a connected directed graph $G=(V,E)$, can be completed in $O(E)$ time.**

However, the Hamiltonian Cycle is NP-Complete cycle.
The traveling salesman problem is a variation of the Hamiltonian cycle.

Polynomial Time Reductions

- **Decision Problems: Problems whose answer is yes or no!!**
Most problems can be converted to decision problems.
- Language recognition problem is a decision problem.
- Suppose $L \subseteq U$ is the set of all inputs for which the answer to the problem is yes –
- L is called the language corresponding to the problem (Turing machines).
The terms language and problem are used interchangeably.
- Given a problem, with an input language X ,
- Now the decision problem can be defined as the problem to recognize whether or not X belongs to L .

Reductions Contd.

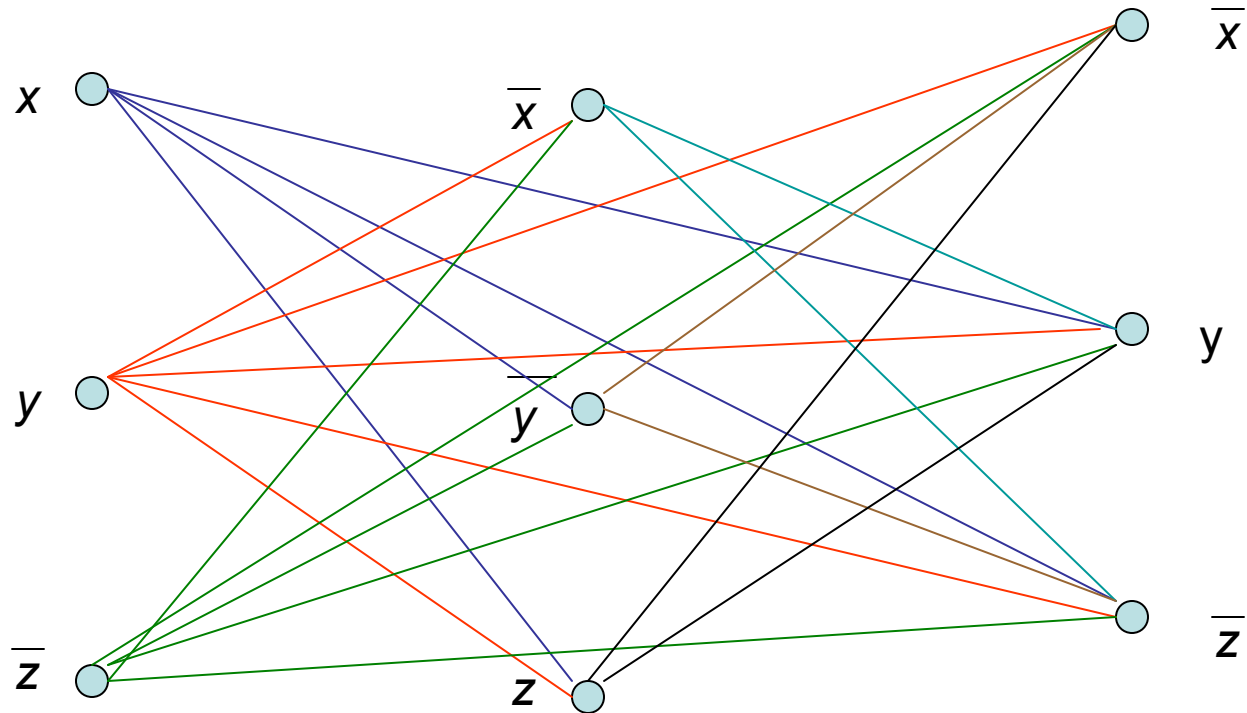
- Definition: Let L_1 and L_2 be two languages from input spaces U_1 and U_2 . We say that L_1 is polynomially reducible to L_2 if there exists a polynomial-time algorithm that converts each input $u_1 \in U_1$ to another input $u_2 \in U_2$ such that $u_1 \in L_1$ iff $u_2 \in L_2$.
- The algorithm is polynomial in the size of the input u_1 .
- If we have an algorithm for L_2 then we can compose the two algorithms to produce an algorithm for L_1 .
- If L_1 is polynomially reducible to L_2 and there is a polynomial-time algorithm for L_2 , then there is a polynomial time algorithm for L_1 .
- Reducibility is not symmetric
- L_1 is polynomially reducible to L_2 does not imply L_2 is polynomially reducible to L_1

The Satisfiability (SAT) Problem

- S – Boolean expression in Conjunctive Normal Form (CNF) (Product (AND) of Sums (ORs))
- For example $S = (x_1+x_2+x_3) \cdot (\bar{x}_1+x_2+x_3) \cdot (\bar{x}_1+\bar{x}_2+x_3)$
- The SAT problem is to determine whether a given Boolean expression is **Satisfiable** (without necessarily finding a satisfying assignment)
- We can guess a truth assignment and check that it satisfies the expression in polynomial time.
- SAT is NP hard
- A Turing machine and all of its operations on a given input can be described by a Boolean Expression. The expression will be satisfiable iff the Turing machine will terminate at an accepting state for the given input.
- <http://www.nada.kth.se/~viggo/problemelist/compendium.html>

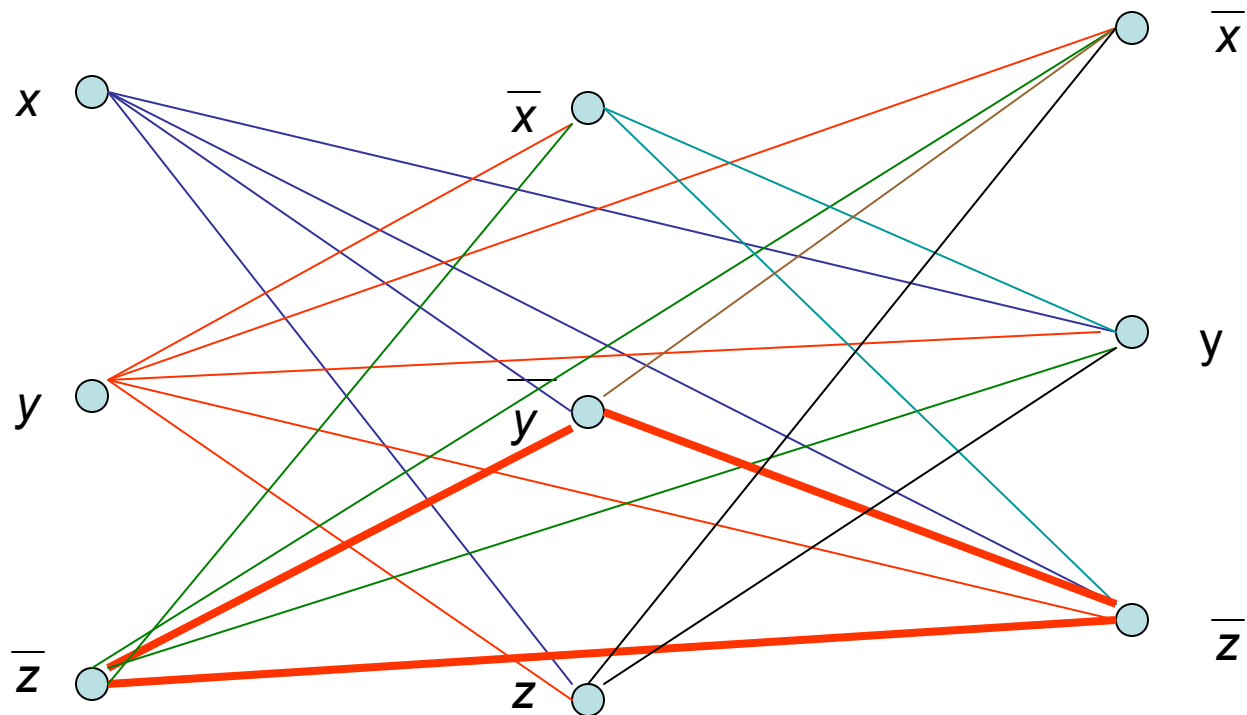
Clique Problem

$$(x + y + \bar{z}) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z})$$



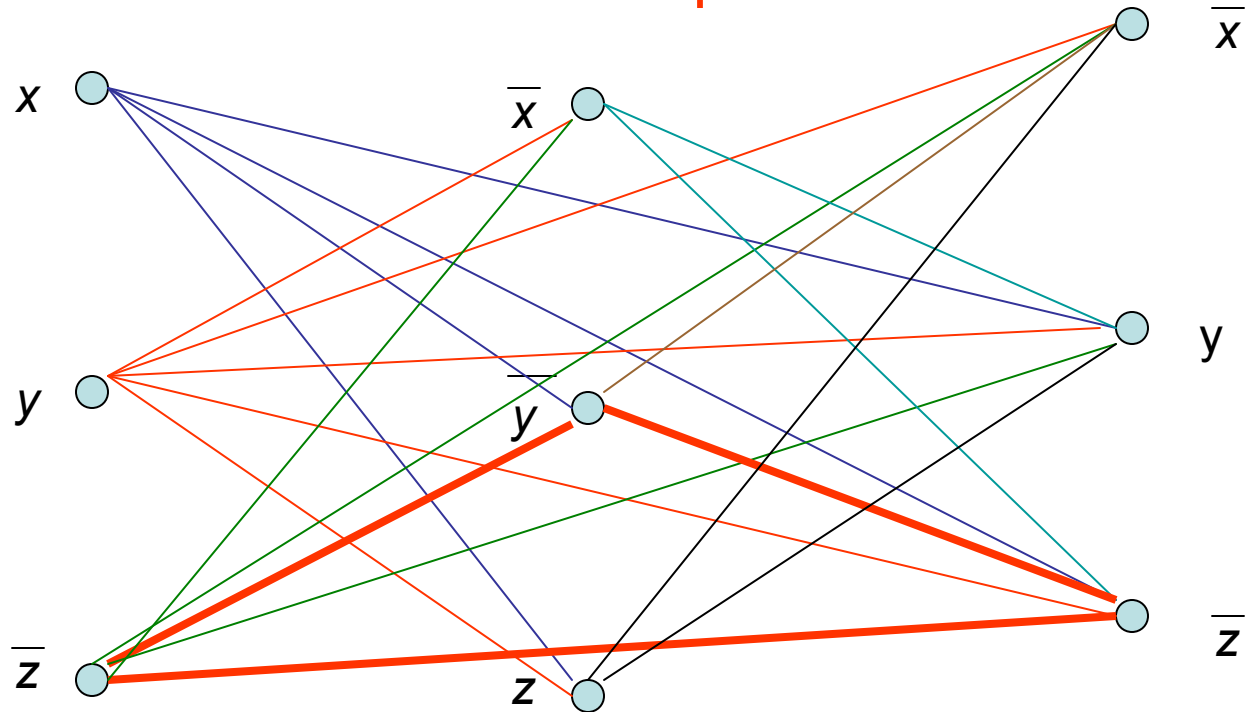
Clique Problem

$$(x + y + \bar{z}) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z})$$



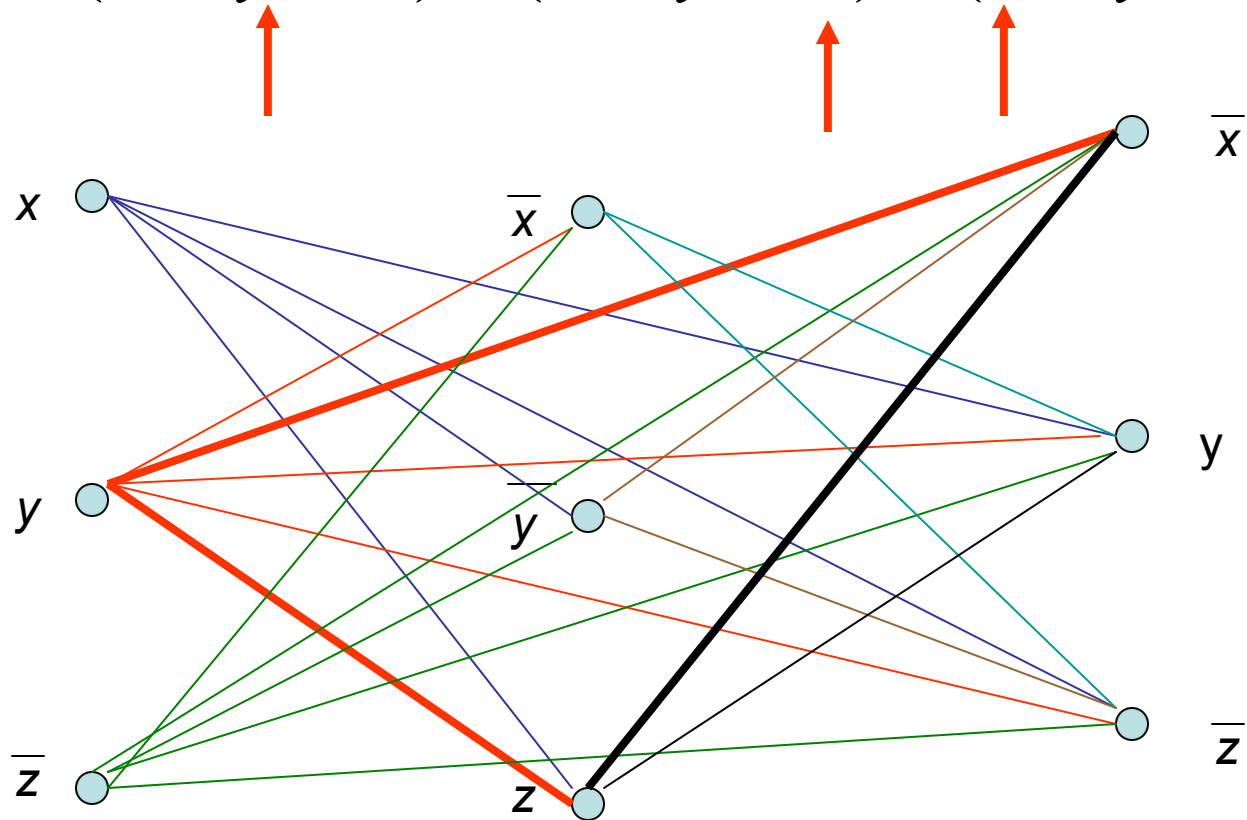
Clique Problem

$$(x + y + \bar{z}) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z})$$



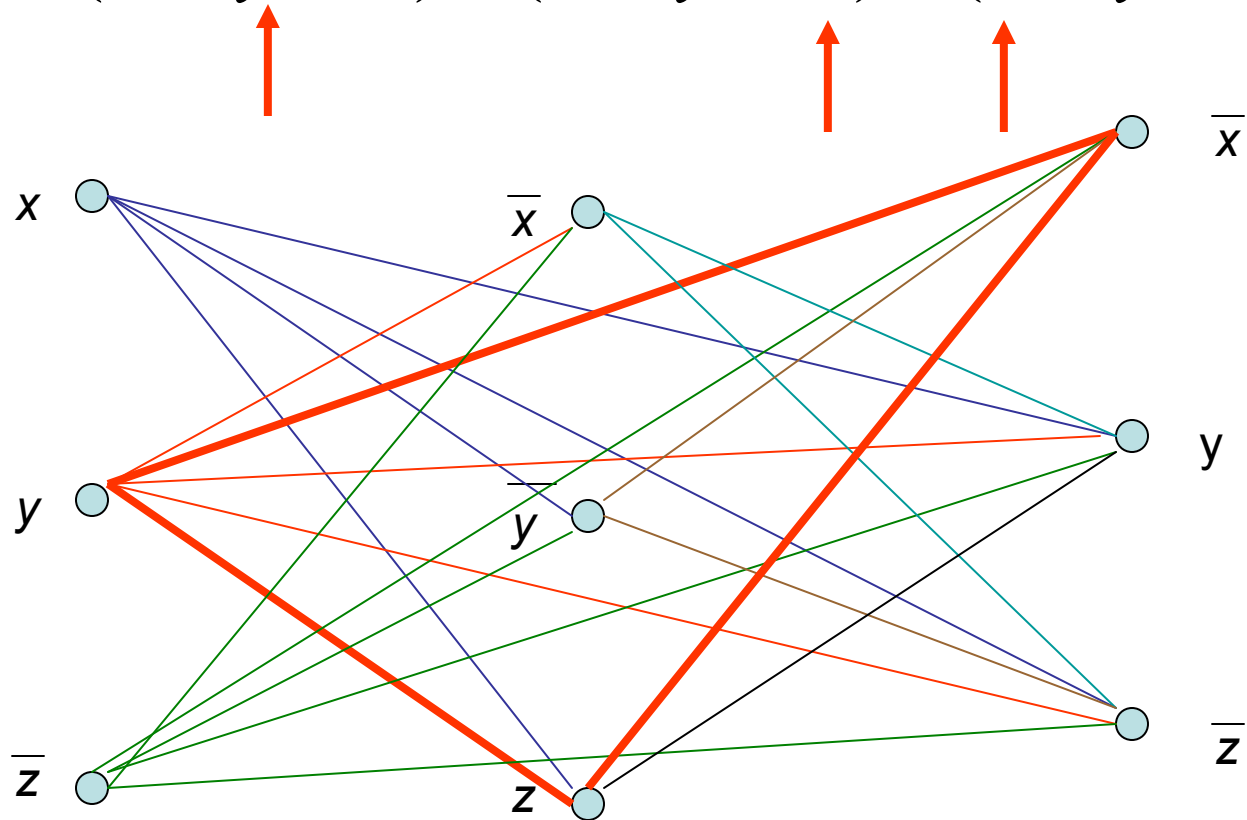
Clique Problem

$$(x + y + \bar{z}) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z})$$



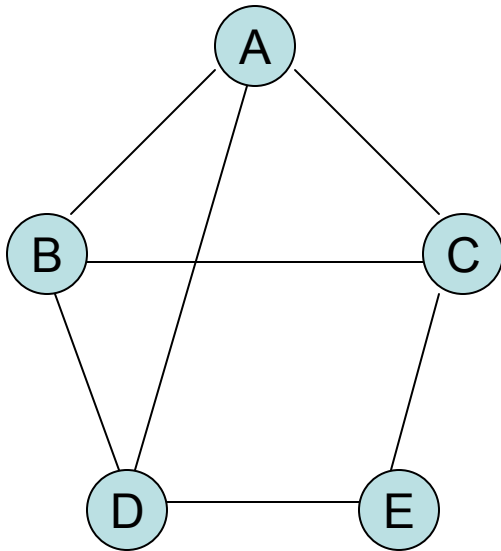
Clique Problem

$$(x + y + \bar{z}) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z})$$



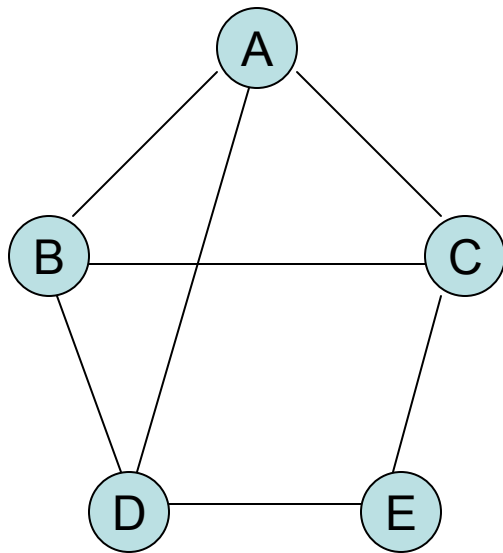
Vertex Cover problem

A vertex cover of $G = (V,E)$ is a set of vertices such that every edge in E is incident to at least one of the vertices in the vertex cover.

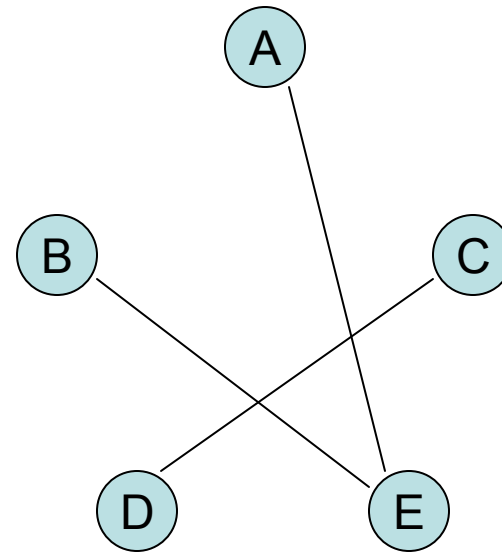


Vertex Cover problem

A vertex cover of $G = (V,E)$ is a set of vertices such that every edge in E is incident to at least one of the vertices in the vertex cover.



G

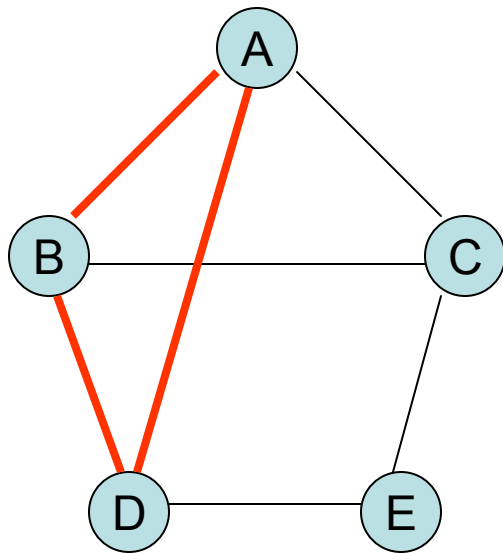


G'

complement of G

Vertex Cover problem

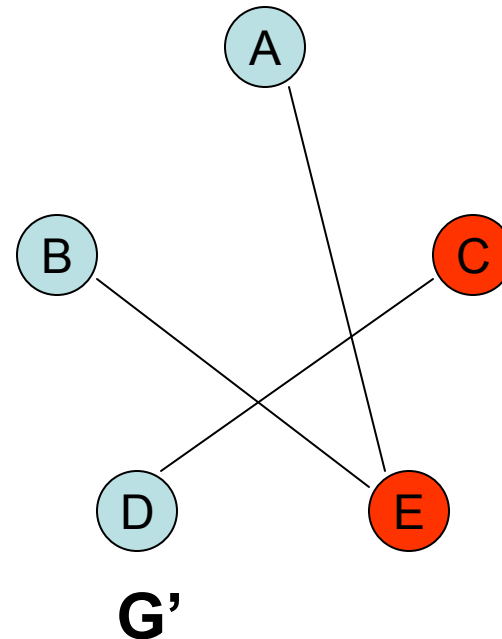
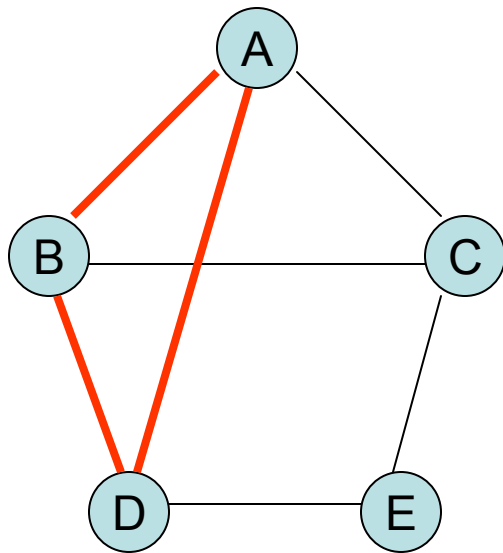
A vertex cover of $G = (V,E)$ is a set of vertices such that every edge in E is incident to at least one of the vertices in the vertex cover.



G has a maximum clique of size $|V| - k$

Vertex Cover problem

A vertex cover of $G = (V,E)$ is a set of vertices such that every edge in E is incident to at least one of the vertices in the vertex cover.

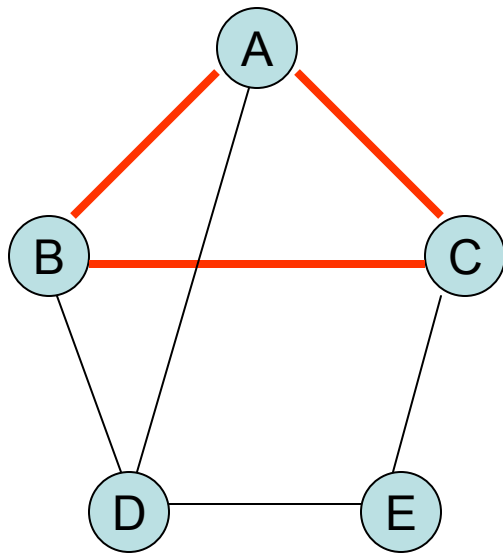


complement of G

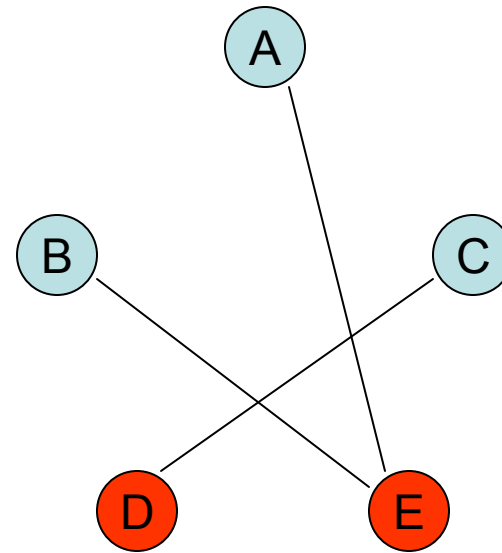
G' has a minimum vertex cover of size k if and only if G has a maximum clique of size $|V| - k$

Vertex Cover problem

A vertex cover of $G = (V,E)$ is a set of vertices such that every edge in E is incident to at least one of the vertices in the vertex cover.



G

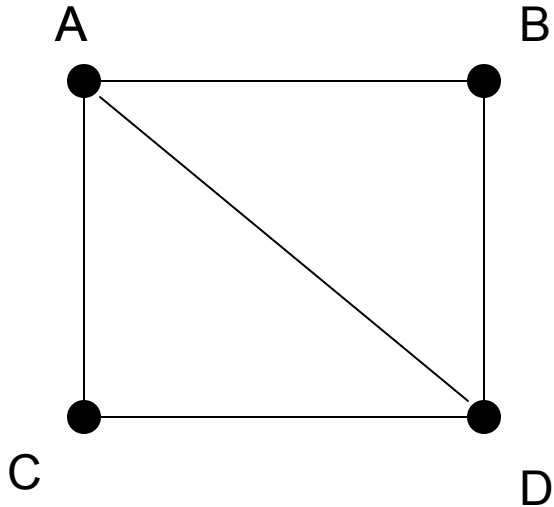


G'

complement of G

Dominating Set

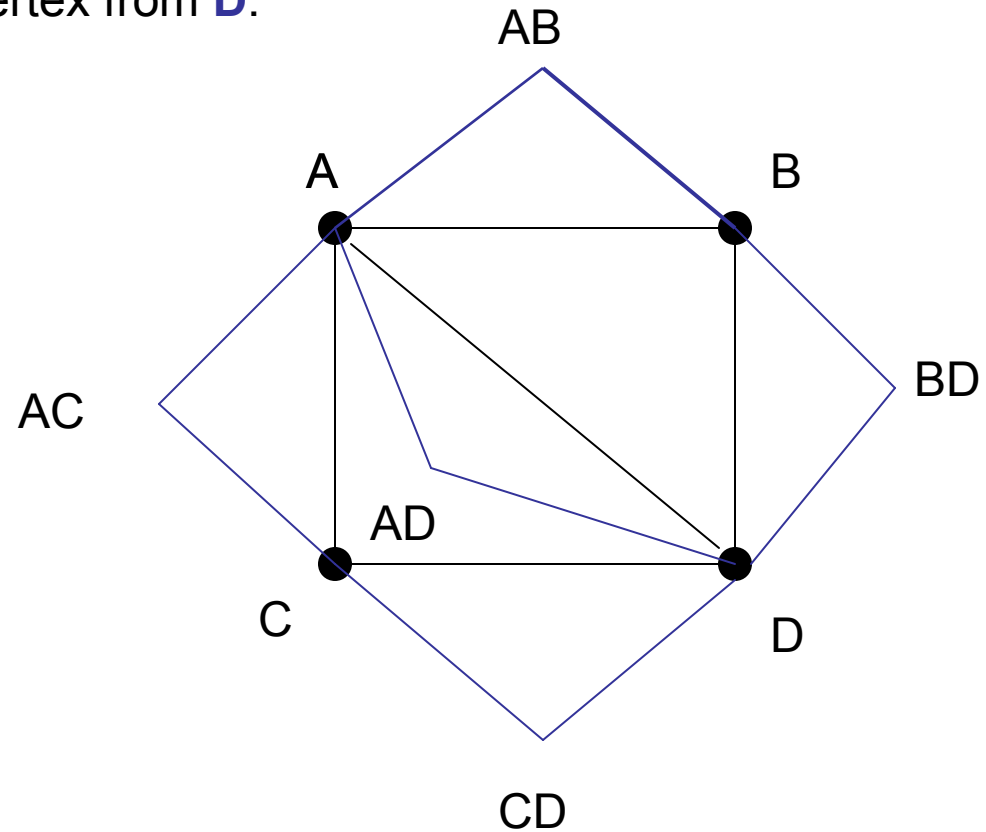
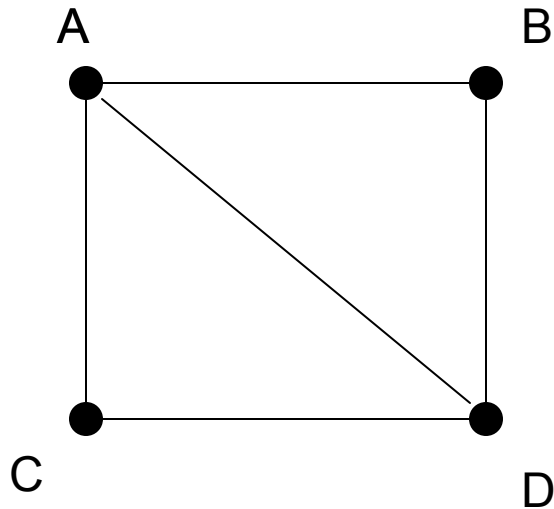
- $G = (V, E)$ is an undirected graph. A dominating set D of G is a set of vertices (a subset of V) such that every vertex of G is either in D or is adjacent to at least one vertex from D .



G has a vertex cover of size m

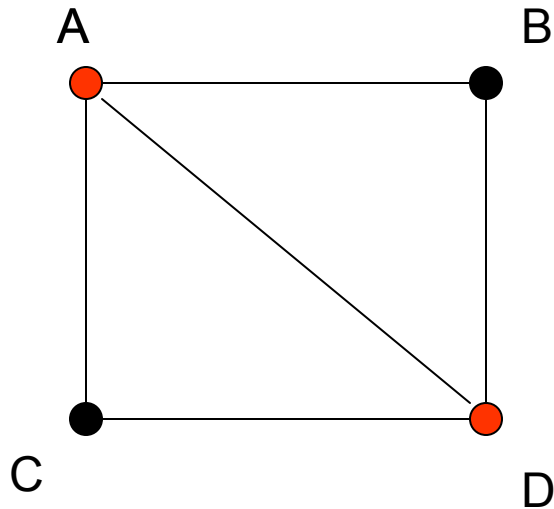
Dominating Set

- $G = (V, E)$ is an undirected graph. A dominating set D of G is a set of vertices (a subset of V) such that every vertex of G is either in D or is adjacent to at least one vertex from D .

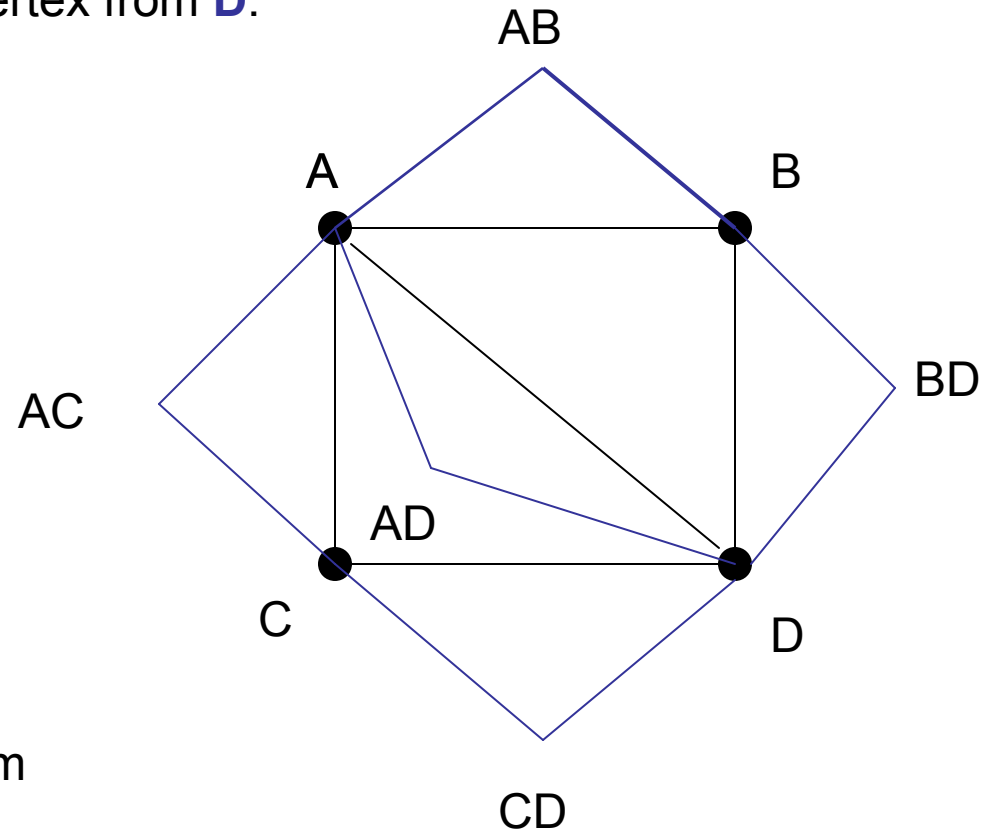


Dominating Set

- $G = (V, E)$ is an undirected graph. A dominating set D of G is a set of vertices (a subset of V) such that every vertex of G is either in D or is adjacent to at least one vertex from D .

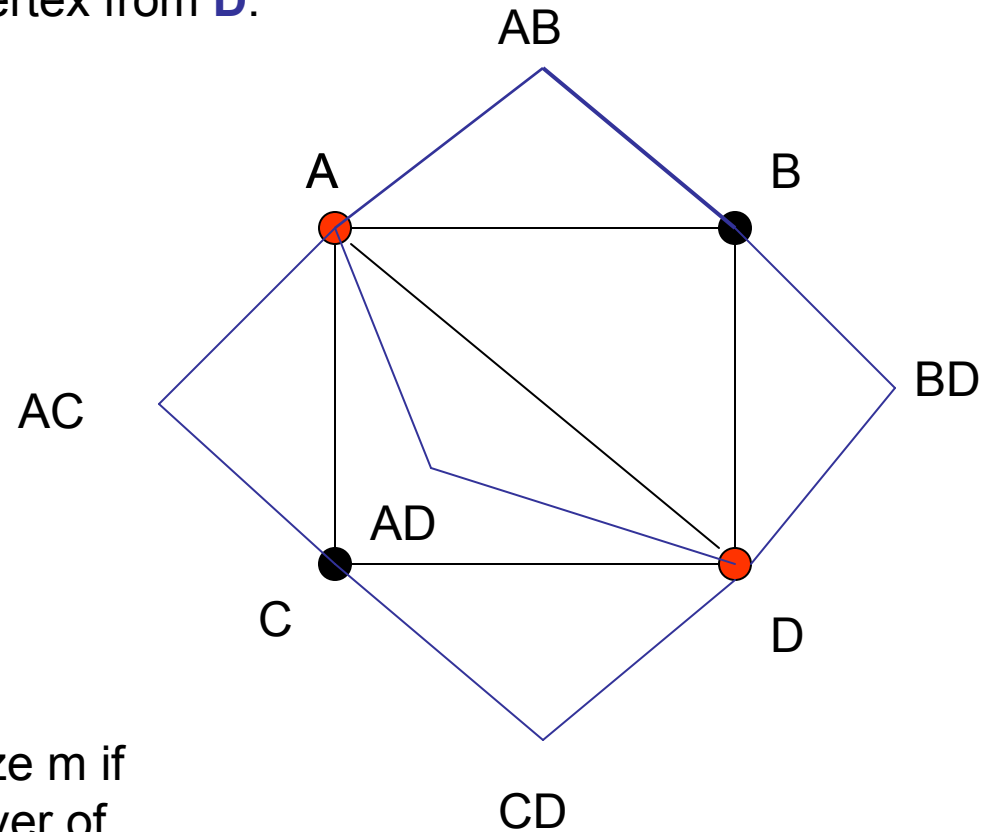
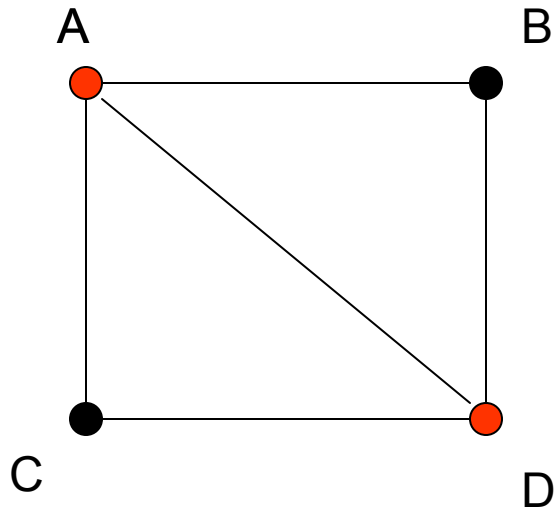


G has a vertex cover of size m



Dominating Set

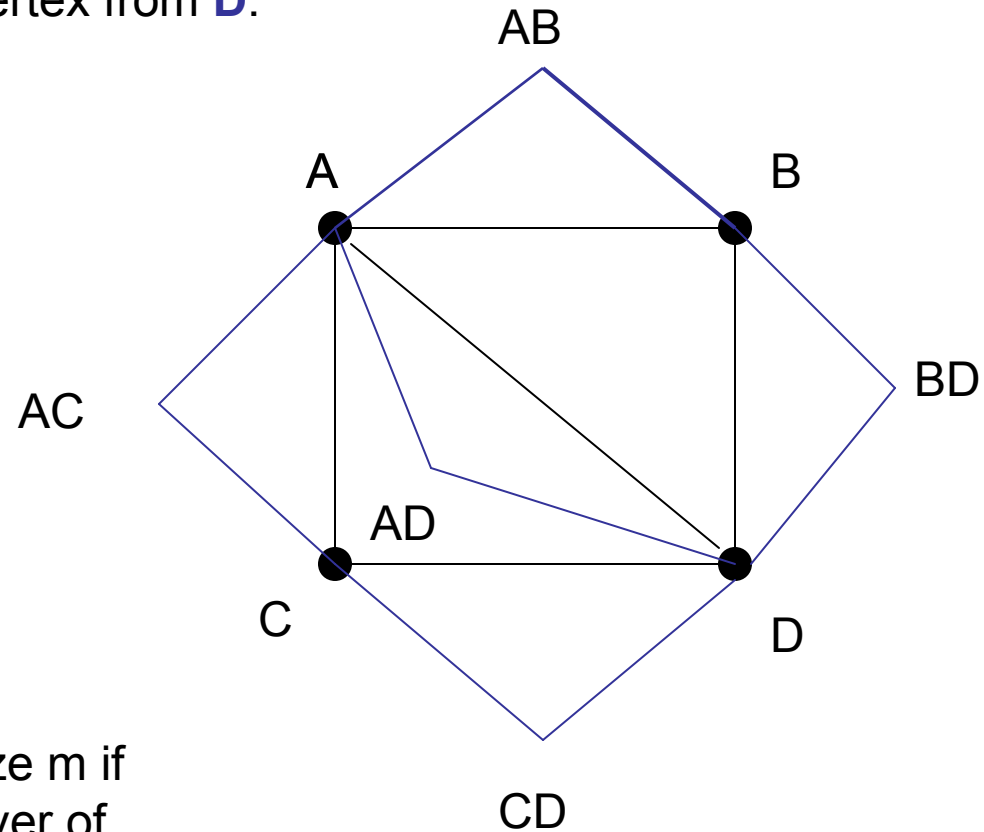
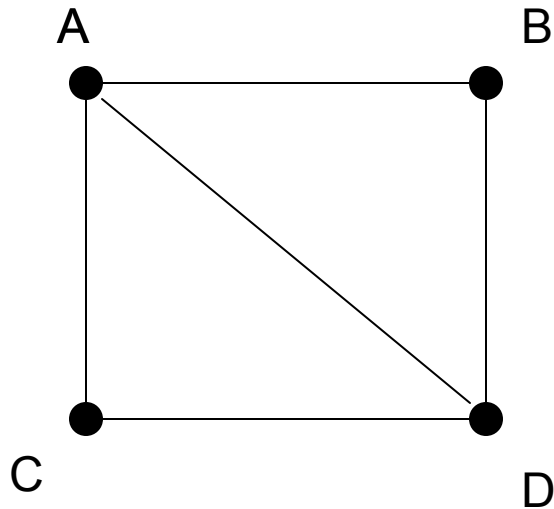
- $G = (V, E)$ is an undirected graph. A dominating set D of G is a set of vertices (a subset of V) such that every vertex of G is either in D or is adjacent to at least one vertex from D .



G' has a dominating set of size m if and only if G has a vertex cover of size m

Dominating Set

- $G = (V, E)$ is an undirected graph. A dominating set D of G is a set of vertices (a subset of V) such that every vertex of G is either in D or is adjacent to at least one vertex from D .



G' has a dominating set of size m if and only if G has a vertex cover of size m

Dealing with NP Complete problems

- Proving that a given problem is NP-Complete does not make the problem go away!! Udi Manber
- An NP-Complete problem cannot be solved precisely in polynomial time
 - We make compromises in terms of optimality, robustness, efficiency, or completeness of the solution.
- Approximation algorithms do not lead to optimal solutions
 - Probabilistic algorithms
 - Branch and bound
 - Backtracking

Backtracking

- The Hamiltonian Circuit Problem
- The Subset problem

Branch and Bound

- Job Assignment
- Knapsack