Course Summary…

Complexity

Tractable Problems

Intractable Problems

Unsolvable Problems

$\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^4)$ NP EXP Undecidable Non-recognizable

Computability

Course Summary…

Complexity

Tractable Problems

Intractable Problems

Unsolvable Problems

$\Theta(n)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n^4)$ NP EXP Undecidable Non-recognizable

Sipser Chap. 7
Cormen Chap. 34

Computability

Polynomial Time

- Most (all?) of the algorithms from the first part of the course run in polynomial time, e.g., $\Theta(n)$, $\Theta(n \log n)$, $\Theta(n^2)$, $\Theta(n^3)$, etc.
- Polynomial time decision problems make up the complexity class P
- If we combine and repurpose solutions that are in P, the result will also be in P

Nondeterministic TMs

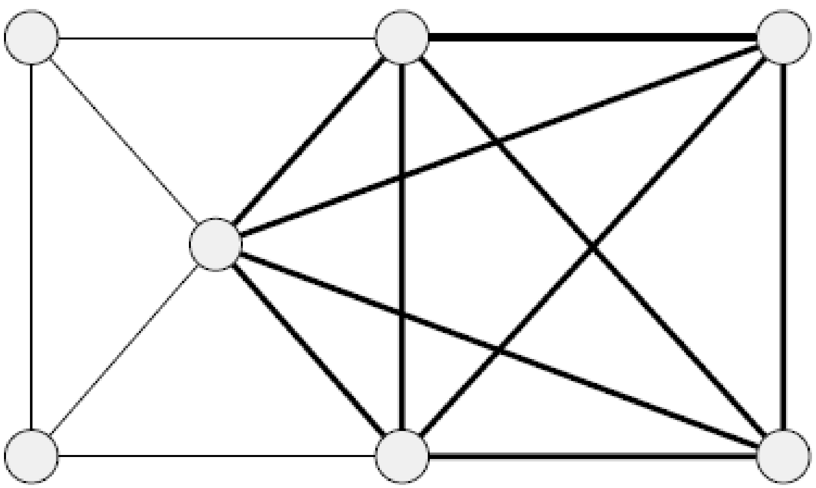
- Nondeterministic Turing Machines (NTMs) are like normal (deterministic) Turing Machines, except that they may have multiple transition rules that can be executed at any given time
- Like NFAs, NTMs will accept as long as there is just one execution sequence that will result in an accept state (others could actually reject!)

Nondeterminism and NP

- Computation time in an NTM is measured by how long it takes to execute a single computation path – we assume that all legal paths can be executed in parallel
- In this way, NTM computation time can be seen as the time it takes to *verify* that a given solution is correct for a given problem
- Problems that can be solved using non-deterministic computation in polynomial time make up the complexity class NP

NP Problem: Clique

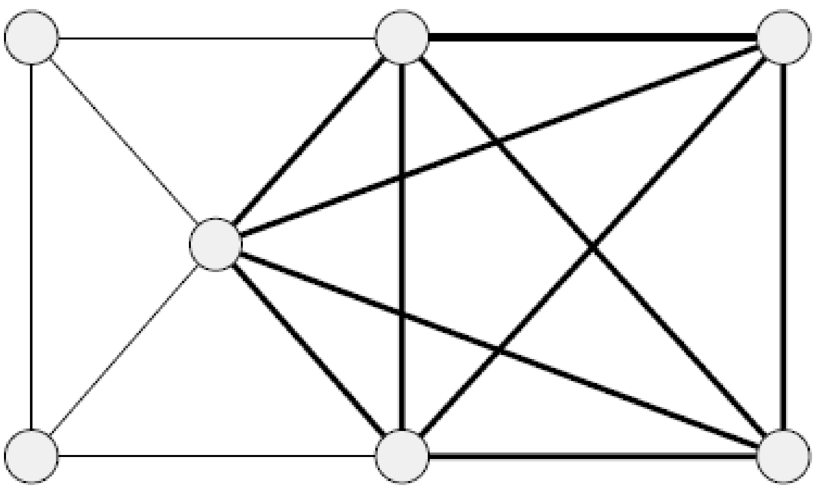
- Given an (undirected) graph G = (V, E), does G have a complete subgraph of k vertices?
  - A complete subgraph (aka, a clique), is where all node pairs are connected via an edge



A graph with a 5-clique

NP Problem: Clique

- There is no known (deterministic) algorithm in P that solves the clique problem
- However, we can find a nondeterministic algorithm could do so in polynomial time



A graph with a 5-clique

NP Problems

- All problems in P are also in NP, in the same way that an $O(n^2)$ algorithm is also $O(n^3)$, or even $O(2^n)$
- Open question: Is it true that $P \neq NP$?
    - If you solve this, you will be the most famous living computer scientist on the planet! (And we'll give you an 'A' in the course!)

NP Complete Problems

- A problem B is said to be NP Complete (NPC) if the following two conditions are met:
  1. B is in NP
  2. Any other NP problem A can be reduced* into the B problem
- NPC problems act as general purpose "solvers" for the class of all NP problems

* i.e., converted in polynomial time

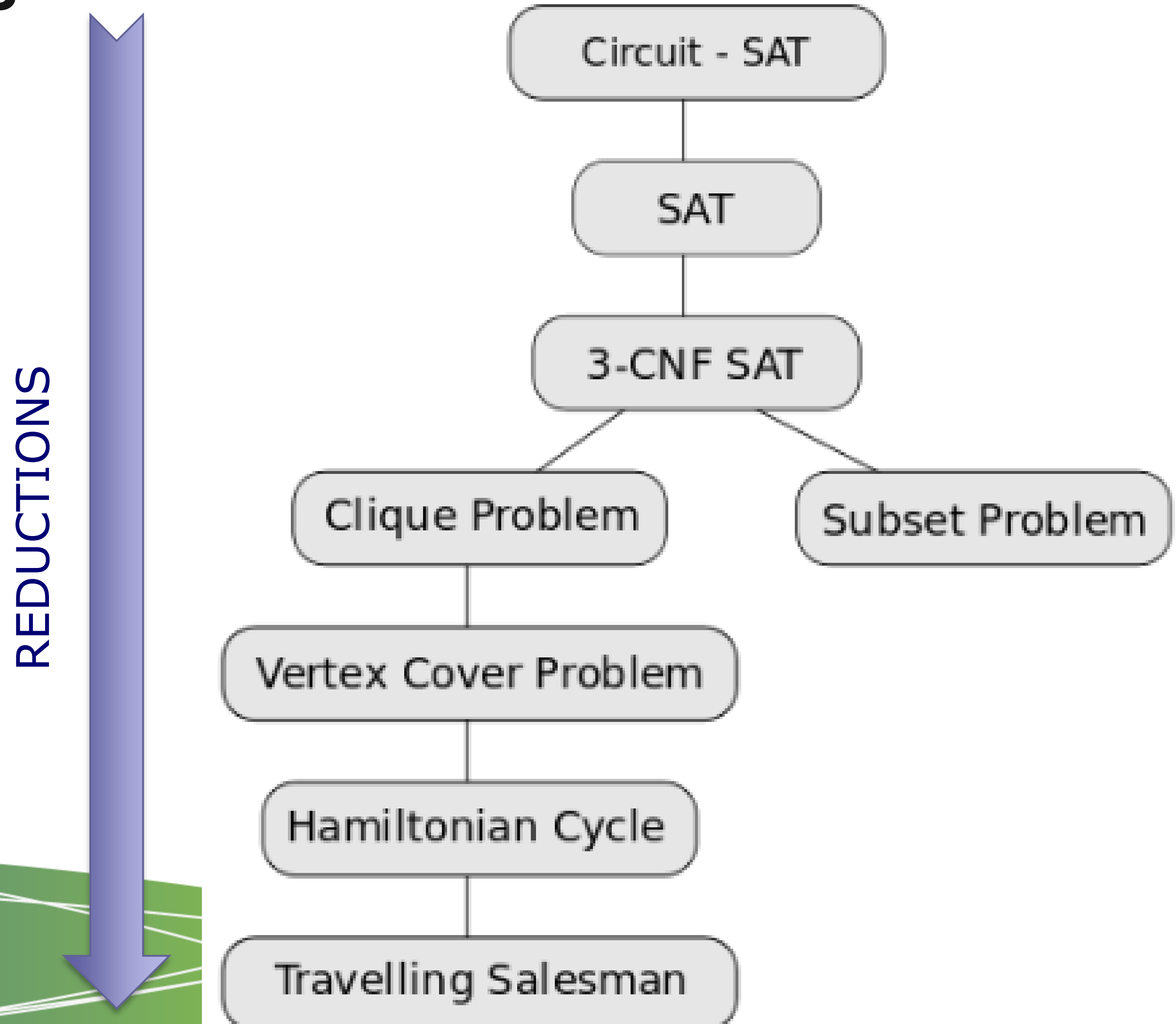
An NPC Problem: SAT

- Given a Boolean formula ϕ constructed using AND, OR, and NOT operators and several Boolean variables, is there a truth assignment to the variables that will make ϕ true?
- Cook-Levin Theorem: SAT is NPC
- Proof Idea: Any NP problem can be converted into a Boolean circuit "computer" that verifies solutions for that problem

Simple Theorem

- If we know that K is NPC, and we can reduce it to another NP problem B, then we know that B is also NPC
  - Proof: by definition of NPC, any NP problem reduces to K, and can be further reduced to B, proving B is also NPC
- This theorem is commonly used to show problems are NPC

Common Reductions Showing NP Completeness



Another Example: 3SAT

- Conjunctive Normal Form (CNF) is where a Boolean formula is a conjunction (ANDs) of disjunctive (OR) clauses containing literals (Boolean variables that may be NOT'ed), e.g.:

$$(x \vee \neg y \vee z) \wedge (y \vee \neg z) \wedge (w \vee \neg x \vee z)$$

- A 3CNF formula is where all of the d-clauses only have 3 literals
- The 3SAT problem is SAT, limited to 3CNF formulas

Reduction Example: 3SAT to Clique

- Construct a graph G where:
  - Each literal in the 3CNF formula represents a node in the graph, where we group the nodes into their respective d-clauses
  - Add an edge between each pair of nodes where:
    - The nodes are not part of the same d-clause
    - One node does not represent the negation of the other

Reduction Example: 3SAT to Clique

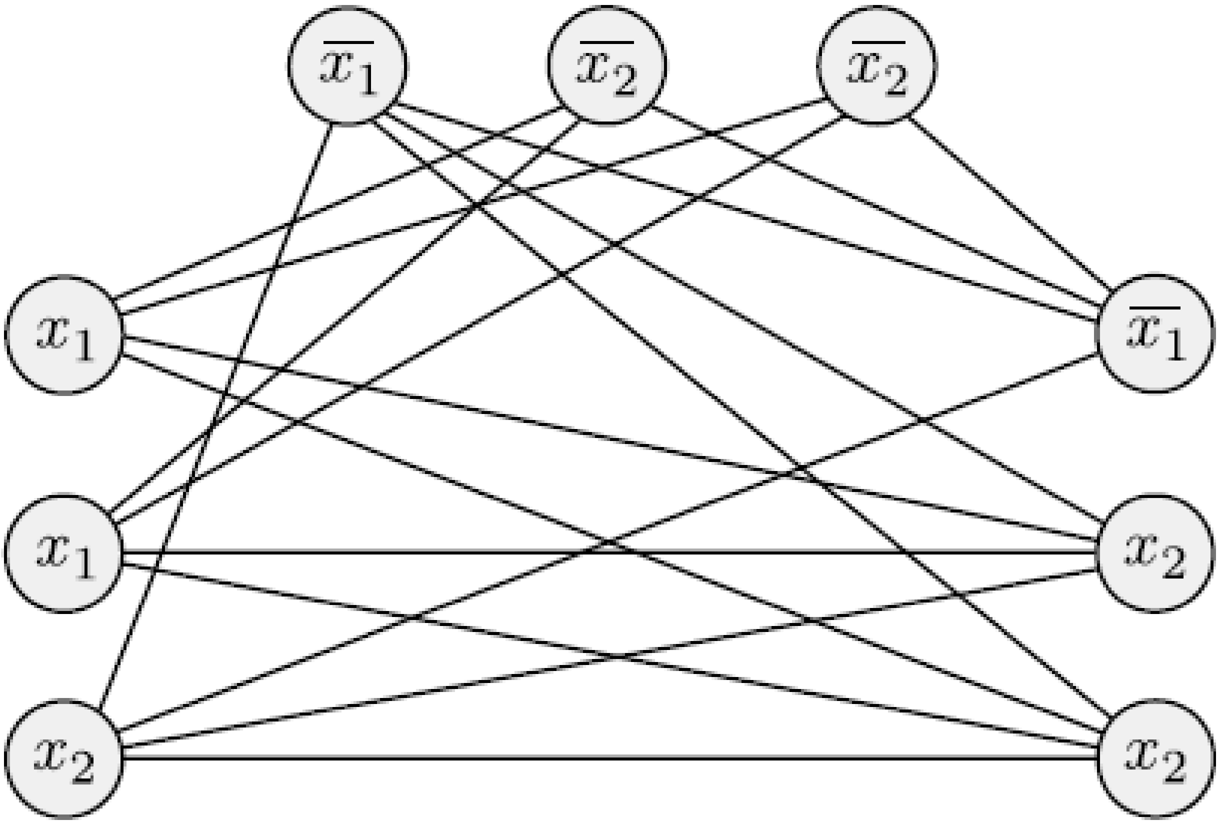


FIGURE 7.33

The graph that the reduction produces from

$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$

Reduction Example: 3SAT to Clique

- Suppose k is the number of d-clauses in our 3CNF formula
- Claim: We can find a k-clique on G iff the formula is satisfiable
- Justification:
  - A k-clique on G must contain exactly one node from each d-clause group, which represents a true literal that can make the whole d-clause true
  - A literal and its negation cannot both have nodes in the k-clique