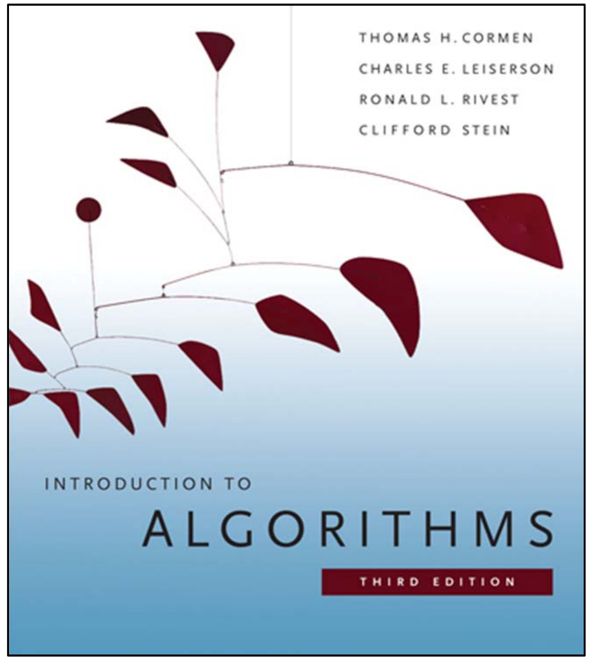
6.006

Introduction to Algorithms



Lecture 25: Complexity

Prof. Erik Demaine

Today

- Reductions between problems
- Decision vs. optimization problems
- Complexity classes
  - P, NP, co-NP, PSPACE, EXPTIME, …
- NP-completeness

$P \stackrel{?}{=} NP$

Reductions

balsamic reduction
http://www.allthingsolive.ca/2011/01/pecorino-di-fossa-cheese-pears-and-cinnamon-pear-balsamic/

design by Gary Anderson
http://en.wikipedia.org/wiki/File:Recycle001.svg

http://cdn.zedomax.com/blog/wp-content/uploads/2010/02/reuse_reduce_recycle.jpg

RECALL: LECTURE 1

How to Design an Efficient Algorithm?

7 EASY STEPS!

1. Define computational problem
2. Abstract irrelevant detail
3. Reduce to a problem you learn here (or 6.046 or algorithmic literature)
4. Else design using “algorithmic toolbox”
5. Analyze algorithm’s scalability
6. Implement & evaluate performance
7. Repeat (optimize, generalize)

Reductions

- Instead of solving a problem from scratch, convert your problem into a problem you already know how to solve
- <u>Examples:</u>
  - Min-product path → shortest path *(take logs)*
  - Longest path → shortest path *(negate weights)*
  - Min multiple-of-5 path → shortest path *($\approx G^5$)*
  - Unweighted → weighted shortest path *(weight 1)*
  - 2D path planning → shortest path *(visibility graph)*

Polynomial-Time Reductions

- Consider two problems A & B
- ***Polynomial-time reduction*** $A \to B$:
  - Solution to A using solution to B
  - Polynomial-time algorithm for A, with free calls to subroutine to solve B
- Write $A \leq_P B$: "A is no harder than B" (up to polynomial overhead)

polynomial: good
exponential: bad

One-Call Reductions

- Common polynomial-time reduction $A \to B$:
  1. Given input to problem A
  2. Polynomial-time preprocessing
  3. One call to solve problem B
  4. Polynomial-time postprocessing

- More interesting than “reduce any problem to basic operations in model of computation”
  - Example: sorting reduces to element comparisons

Decision Problems

Hamlet (1948)

- ***Decision problem*** = any problem whose answer is one bit: "yes" or "no"
- <u>Examples:</u>
  - Do these line segments have an intersection?
  - Does this Super Mario Bros. level have a solution?
  - Does the $3 \times 3 \times 3$ Rubik's Cube always have a solution in 20 moves?
  - Given a sequence of cards, is there a Crazy Eights subsequence trick of at least 17 cards?
  - Does given weighted graph have a negative cycle?

Optimization → Decision

- Any optimization problem can be converted into a decision problem
- Add input b: bound on optimal solution
  - Maximization problem ⇒ b is a lower bound
  - Minimization problem ⇒ b is an upper bound
- Examples:
  - Given sequence of cards & number b,
    is there a Crazy Eights trick of $\geq b$ cards?
  - Given a weighted graph, vertices s & t, number b,
    is there an $s \to t$ path of weight $\leq b$?

Why Decision Problems?

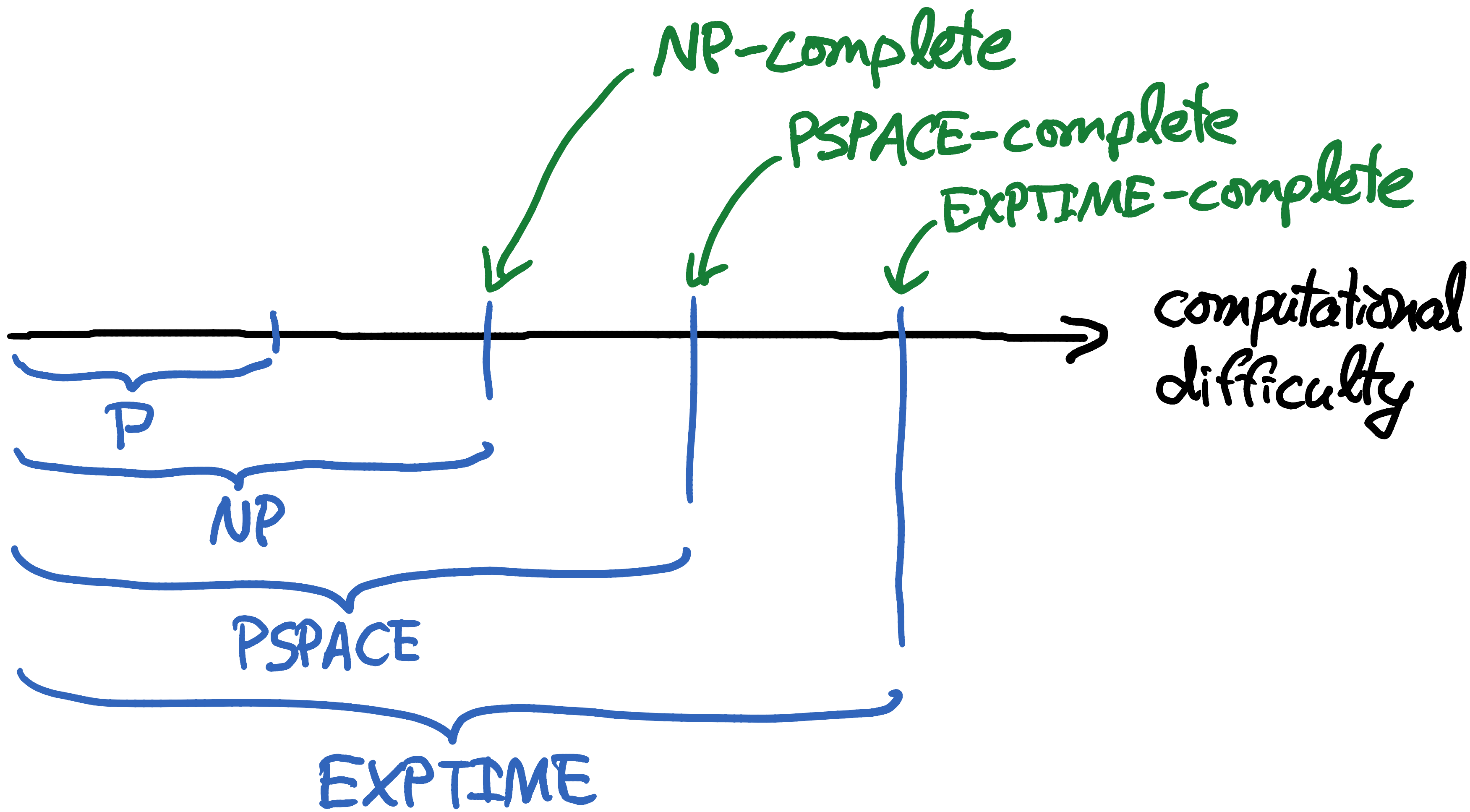
- <u>Meta claim:</u> Every computational problem has a decision version of roughly equal computational difficulty
- Maximization/minimization problems: binary search on bound b to find optimal
  - Logarithmic overhead
- <u>Example:</u> Is key k in this binary search tree?
- <u>Example:</u> Given unsorted $A[1..n]$ and i & r, does $A[i]$ sort to rank $\leq r$ in a sorted array? } $\Theta(n)$

Karp Reductions

Richard Karp

- For decision problems A and B
- Simplest (and most common) type of polynomial-time $A \to B$ reduction:
    1. Given input to problem A
    2. Polynomial-time preprocessing
    3. One call to solve problem B
    4. Return the **same answer** (no postprocessing)

Complexity Classes



Time Classes

- ***P*** = class of all decision problems solvable by polynomial-time algorithms
  - $n, n^2, n^3, \ldots$
- ***EXPTIME*** = class of all decision problems solvable by exponential-time algorithms
  - $2^n, 2^{n^2}, 2^{n^3}, \ldots$
- In general, if $f(n) = o(g(n)/\lg n)$, then $\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(g(n))$
  (Time Hierarchy Theorem)

Space Classes

- ***PSPACE*** = class of all decision problems solvable using polynomial storage space
- Example: Is there a solution to a given $n \times n \times n$ Rubik's Cube using $\leq k$ moves?

  – BFS ✗ bad   – DFS with bound on depth ✓
- ***EXPSPACE*** = class of all decision problems solvable using exponential storage space
- **Space Hierarchy Theorem:** if $f(n) = o(g(n))$, then $\mathrm{SPACE}(f(n)) \subsetneq \mathrm{SPACE}(g(n))$

$\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$

NP

Nondeterministic Polynomial time

- ***NP*** = class of all decision problems solvable by a "lucky" polynomial-time algorithm
  - In $O(1)$ time, can ***guess*** between two choices
  - At the end, report "yes" or "no"
  - If any way to say "yes", actually return "yes" (always make right choice)
- Example: Is there a solution to a given $n \times n \times n$ Rubik's Cube using $\leq k$ moves?
  - Guess first move, second move, … , kth move
  - Return "yes" if solved

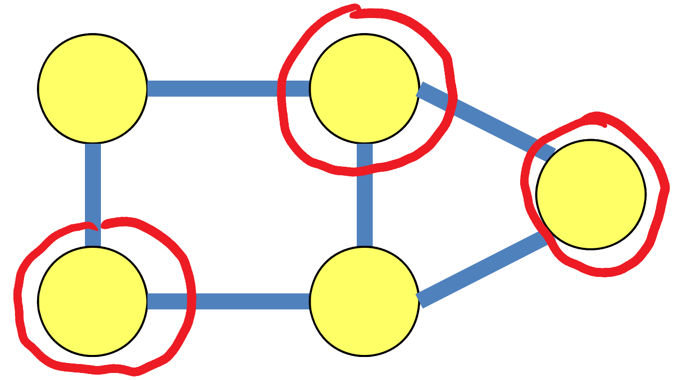
NP Remix

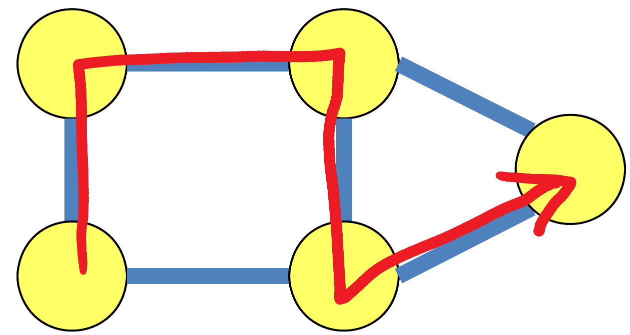
- Equivalently, ***NP*** = class of all decision problems ***verifiable*** in polynomial time
  - Every "yes" input has a polynomial-length ***certificate*** *(might be very hard to find)*
  - Given input and certificate, can confirm that answer is "yes" in polynomial time
- Example: Is there a solution to a given $n \times n \times n$ Rubik's Cube using $\leq k$ moves?
  - Certificate = sequence of $\leq k$ moves to solution

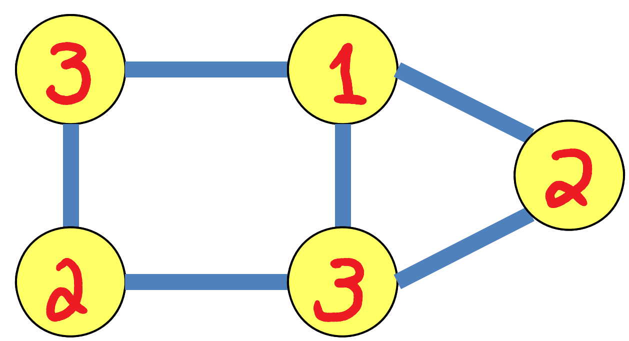
NP Problems

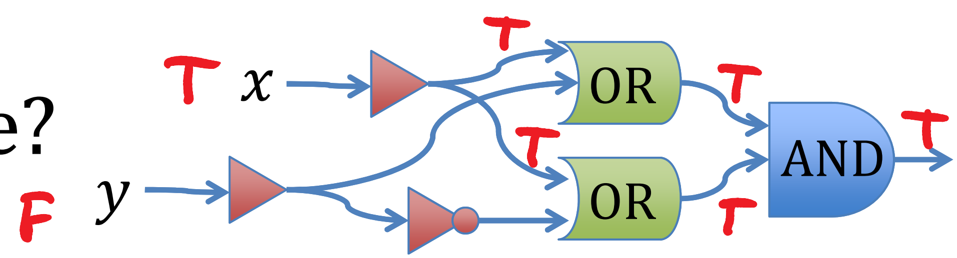
- Given a graph, does it have vertex cover of size $\leq k$?
- Given a graph, is there a simple path of length $\geq k$?
- Given a graph, can it be colored with 3 colors?
- Given a Boolean circuit, are there inputs that make the output true?

(SAT)

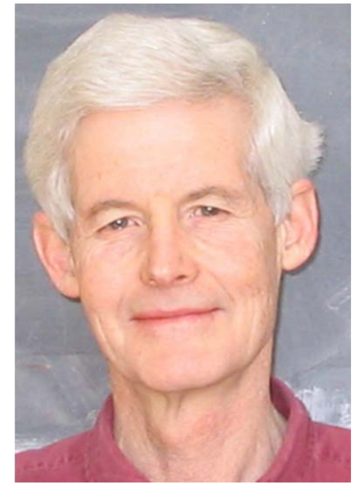








NP-Completeness

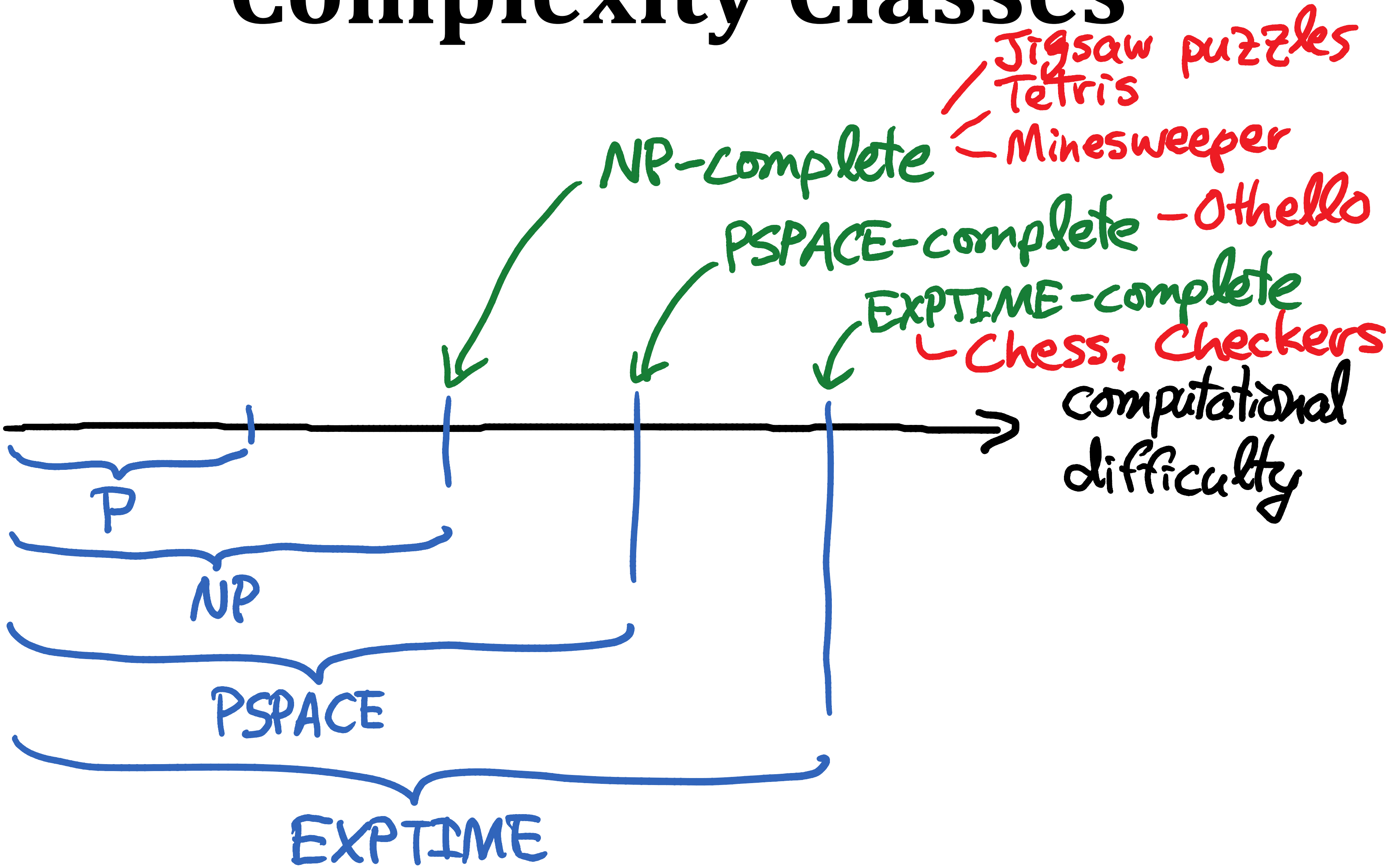


Stephen Cook

- ***NP-complete*** = NP problem that is at least as hard as all problems in NP
  - Formally: $B \in \text{NP}$ and $A \leq_P B$ for all $A \in \text{NP}$
  - Hardest problems in NP, all essentially equivalent

- If there are any problems in $\text{NP} - \text{P}$, then NP-complete problems are among them
- If any NP-complete problem has a polynomial-time algorithm, then $\text{P} = \text{NP}$

Complexity Classes

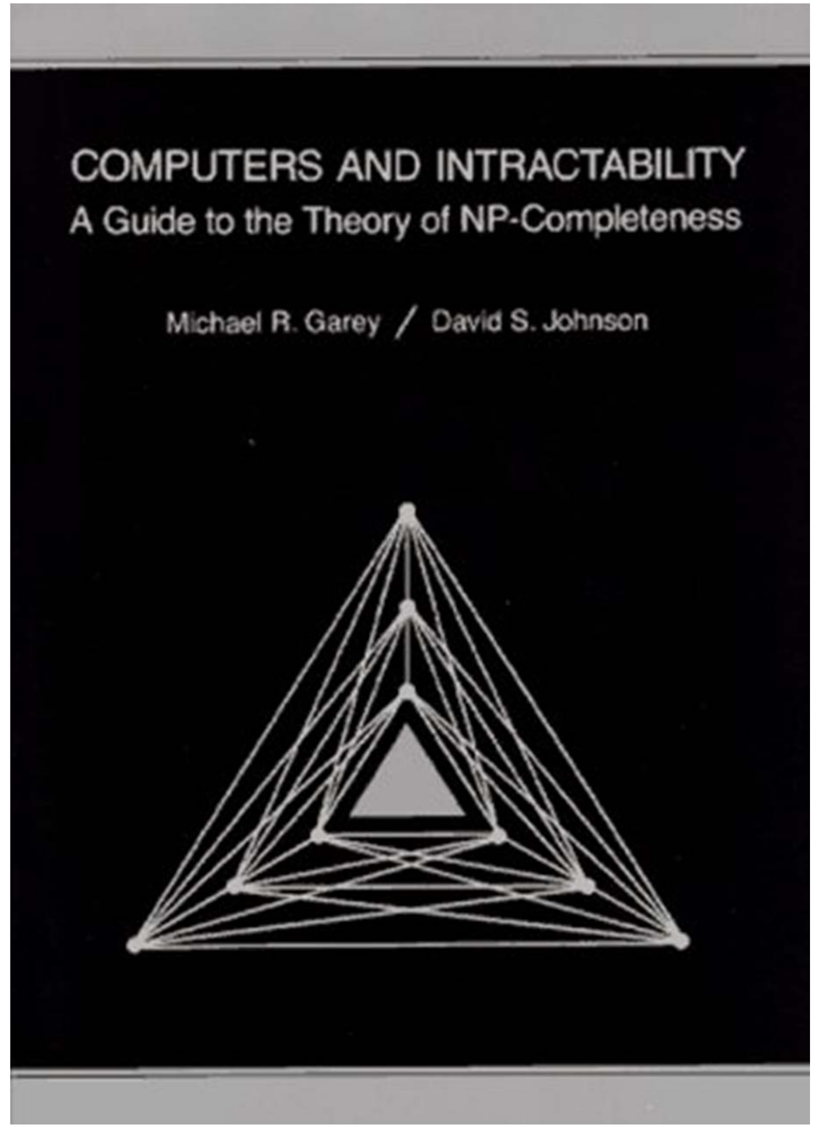


Power of Reduction

- If A is NP-complete, $A \leq_P B$, and $B \in$ NP, then B is NP-complete
- Proof:
  - $C \leq_P A$ for all $C \in$ NP
  - $A \leq_P B$
  - $C \leq_P B$ for all $C \in$ NP

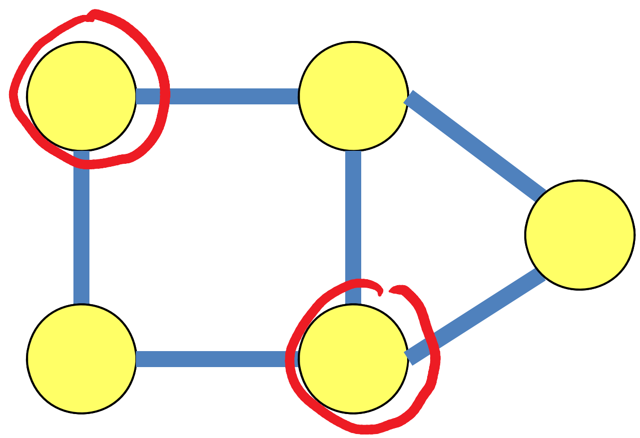
Proving NP-Completeness

- Start with any known NP-complete problem A
  - Given a graph, does it have vertex cover of size $\leq k$?
  - Given a graph, is there a simple path of length $\geq k$?
  - Given a graph, can it be colored with 3 colors?
  - Given a Boolean circuit, are there inputs that make the output true?
- Prove $A \leq_P B$ and $A \in$ NP



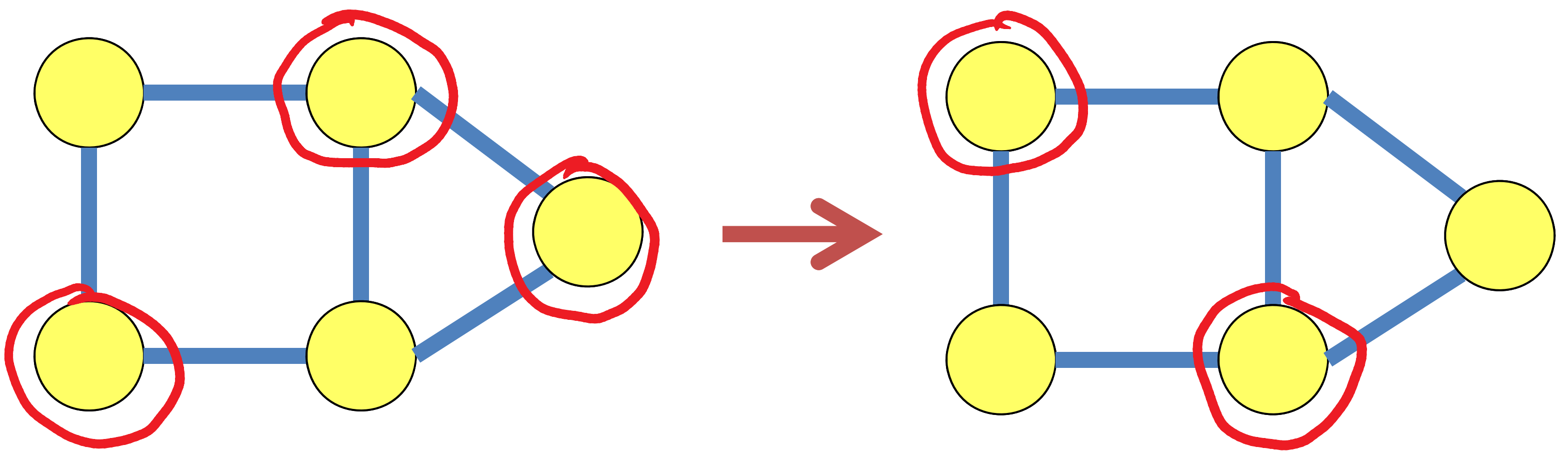
Independent Set

- ***Independent set*** =
  subset of vertices inducing no vertices
- <u>Problem:</u> Given a graph G and integer k, is there an independent set of size $\geq k$?



Vertex Cover $\leq_P$ Independent Set

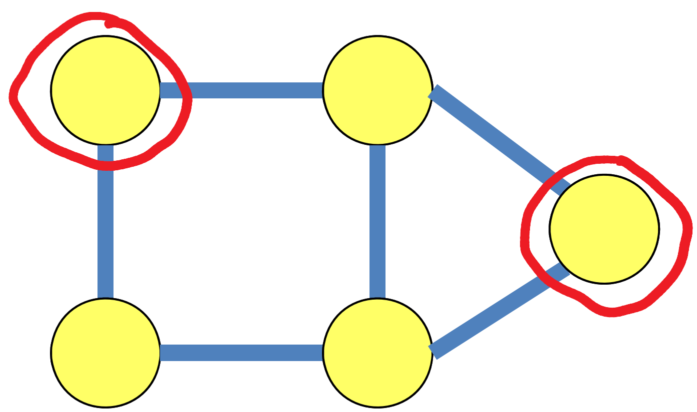
- G has a vertex cover of size $\leq k$
  if and only if
  G has an independent set of size $\geq |V| - k$
- So $(G, k) \mapsto (G, |V| - k)$ reduces VC $\rightarrow$ IS
- VC is NP-complete $\Rightarrow$ IS is NP-complete



Vertex Cover   Independent Set

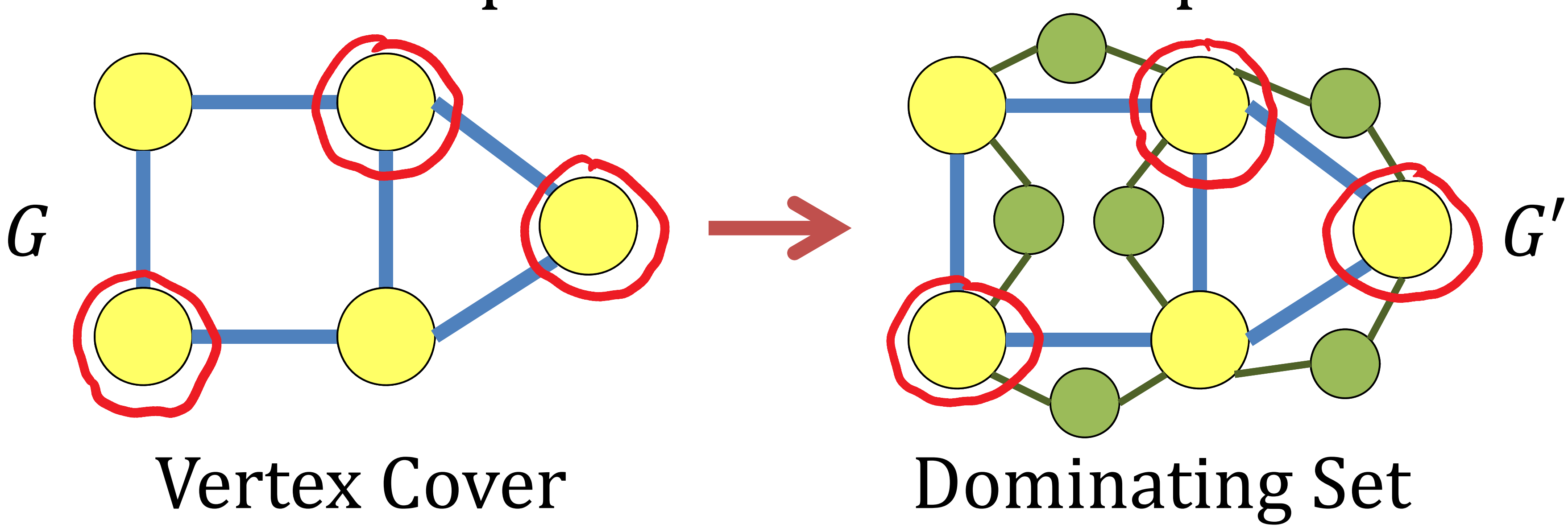
Dominating Set

- ***Dominating set*** = subset of vertices such that every other vertex is adjacent to someone in the subset
- Problem: Given a graph G and integer k, is there a dominating set of size $\leq k$?



Vertex Cover $\leq_P$ Dominating Set

- G has a vertex cover of size $\leq k$
  if and only if
  G' has a dominating set of size $\leq k$
- So $(G, k) \mapsto (G', k)$ reduces VC $\rightarrow$ DS
- VC is NP-complete $\Rightarrow$ DS is NP-complete



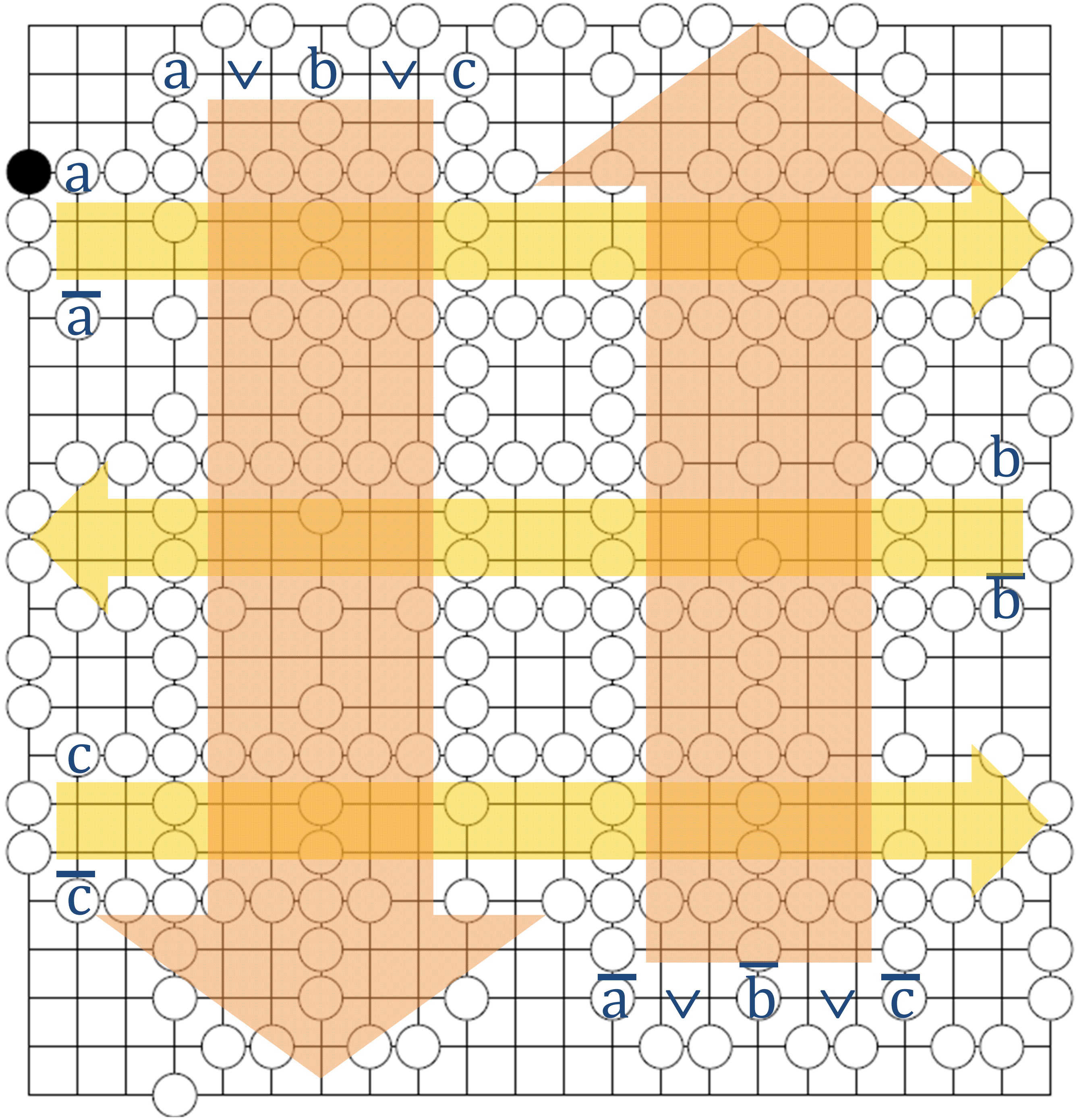
Phutball

[Conway]

“Mate in 1” (can I win in one move?) NP-complete

[Demaine, Demaine, Eppstein 2000]

$$(a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee \overline{c})$$



Bad & Good News

- Many problems are NP-complete
- Can often find approximate solutions in polynomial time
  - Within $O(1)$ factor of optimal (e.g., Vertex Cover)
  - Within 0.0001% of optimal (e.g., Vertex Cover in planar graphs)