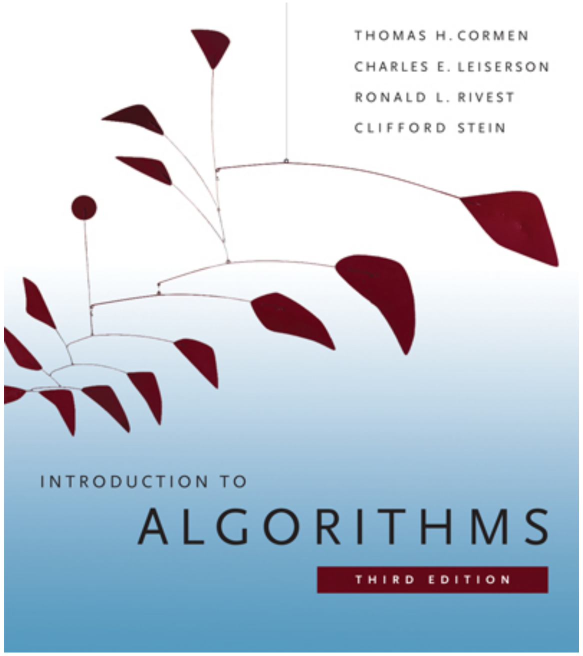
# 6.006- *Introduction to Algorithms*




## *Lecture 24*

**Prof. Patrick Jaillet**

# Outline

- Decision vs optimization problems
- P, NP, co-NP
- Reductions between problems
- NP-complete problems
- Beyond NP-completeness

**Readings CLRS 34**

# Decision problems

- A ***decision problem*** asks us to check if something is true (possible answers: ‘yes’ or ‘no’)
- Examples:
  - PRIMES
    - Instance: A positive integer $n$
    - Question: is $n$ prime?
  - COMPOSITES NUMBERS
    - Instance: A positive integer $n$
    - Question: are there integers $k>1$ and $p>1$ such that $n=kp$?

# Optimization problems

- An ***optimization problem*** asks us to find, among all feasible solutions, one that maximizes or minimizes a given objective
- Example:
  - single shortest-path problem
    - Instance: Given a weighted graph G, two nodes s and t of G
    - Problem: find a simple path from s to t of minimum total length
  - Possible answers: 'a shortest path from s to t ' or 'no path exists between s and t'.

# Decision version of an optimization

- A ***decision version*** of a given ***optimization problem*** can easily be defined with the help of a bound on the value of feasible solutions

- Previous example:
  - SINGLE SPP
    - Instance: A weighted graph G, two nodes s and t of G, and a bound b
    - Question: is there a simple path from s to t of length at most b?

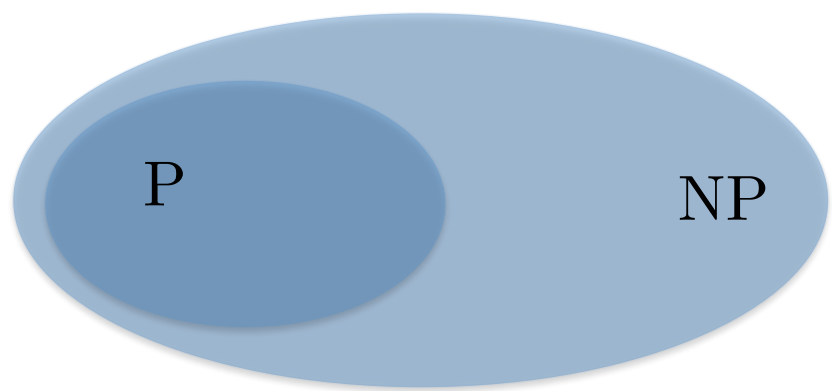
# Optimization vs Decision version

- Clearly, if one can solve an optimization problem (in polynomial time), then one can answer the decision version (in polynomial time)
- Conversely, by doing binary search on the bound $b$, one can transform a polynomial time answer to a decision version into a polynomial time algorithm for the corresponding optimization problem

- In that sense, these are essentially equivalent. We will then restrict ourselves to decision problems

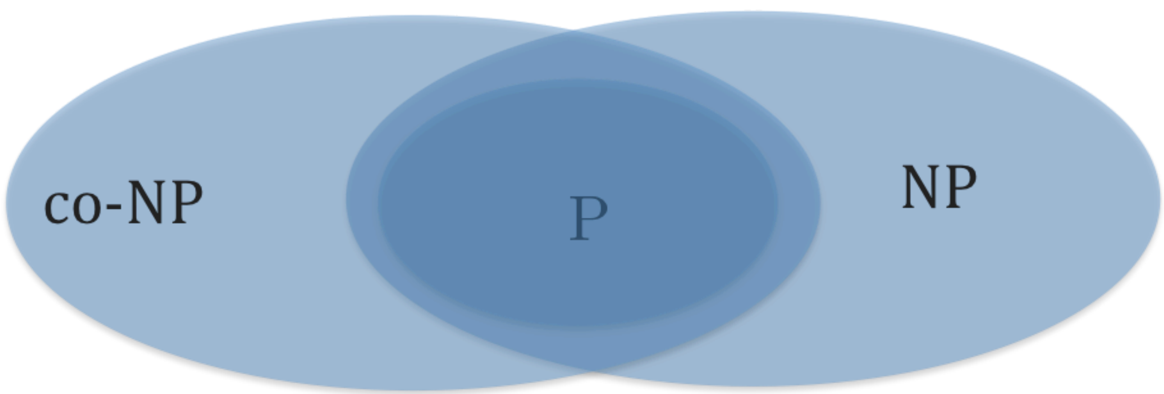
# The classes P and NP

- P is the class of all decision problems that can be solved in polynomial time.
- NP is the class of all decision problems that can be verified in polynomial time:
  - any "yes-instances" can be checked in polynomial time with the help of a short certificate.
- Clearly $P \subseteq NP$



# The class co-NP

- co-NP is the class of all decision problems whose no answers can be verified in polynomial time:
  - any "no-instances" can be checked in polynomial time with the help of a short certificate.
- So clearly $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{co\text{-}NP}$

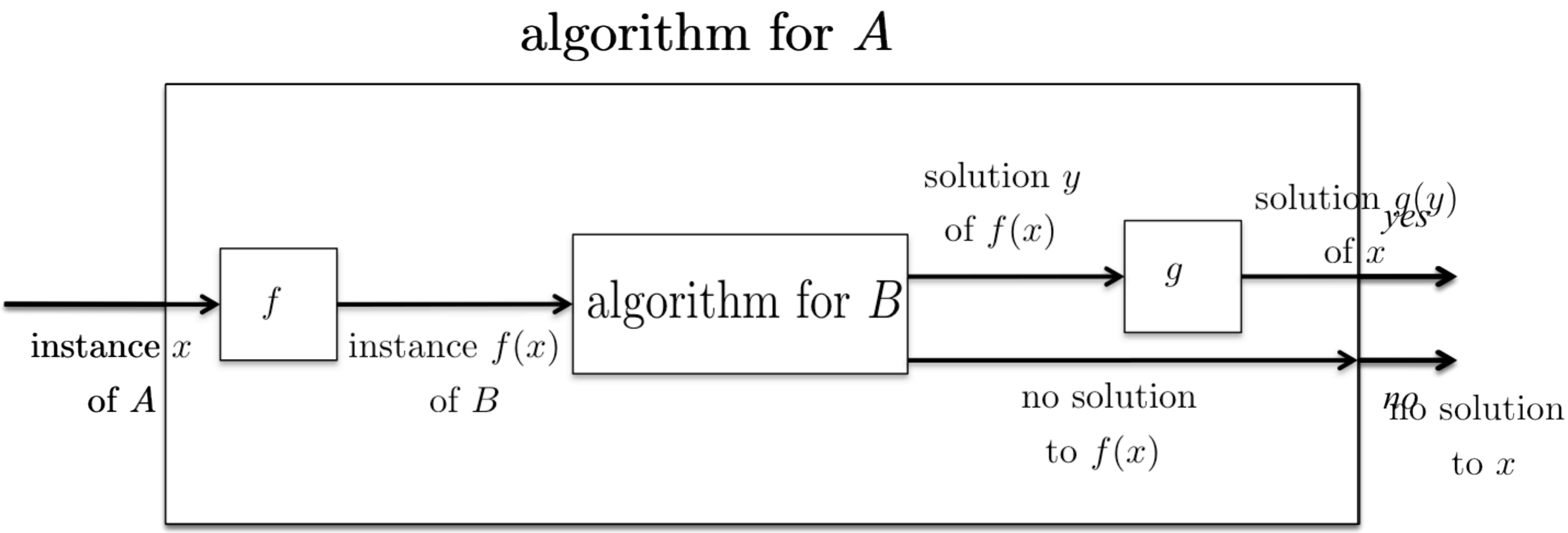


# Reductions between problems

- A polynomial-time reduction from a decision problem A to a decision problem B is a procedure that transforms any instance $I_A$ of A into an instance $I_B$ of B with the following characteristics:
  - the transformation takes polynomial time
  - the answer for $I_A$ is yes iff the answer for $I_B$ is yes
- We say that $A \leq_P B$

# Reductions between problems

- if $A \le_P B$, then one can turn an algorithm for B into an algorithm for A:



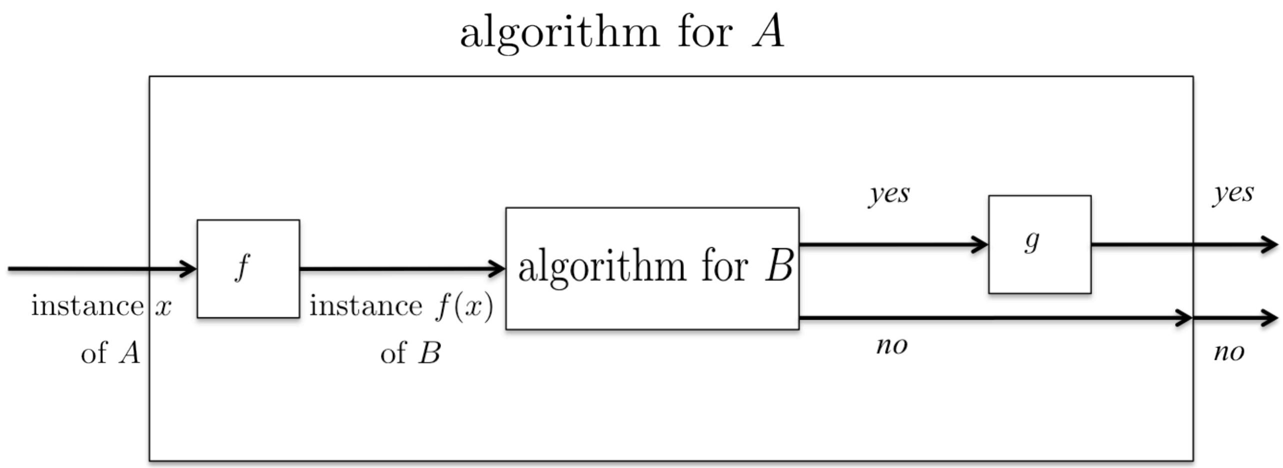

- Reductions are of course useful for optimization problems as well

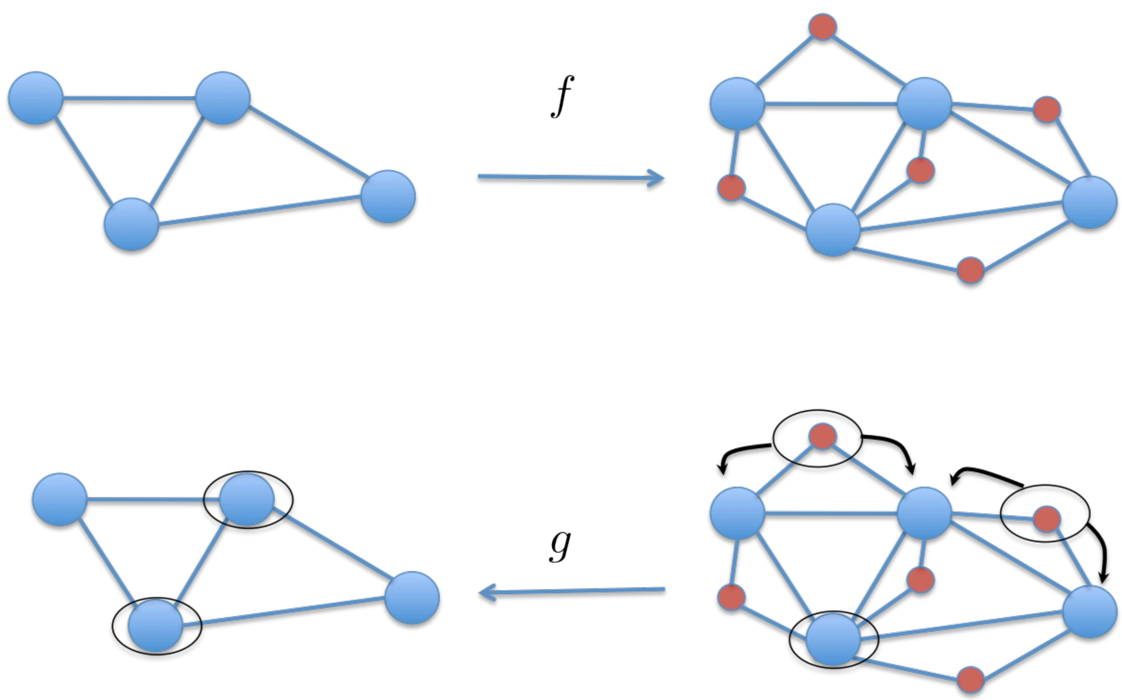
# VERTEX-COVER $\leq_P$ DOMINATING SET

- VERTEX-COVER
  - Instance: a graph G and a positive integer k
  - Question: is there a *vertex cover* (i.e. set of vertices "covering" all edges) of size k or less?

- DOMINATING SET
  - Instance: a graph G and a positive integer p
  - Question: is there a *dominating set* (i.e. set of vertices "covering" all vertices) of size p or less?

# VERTEX-COVER $\leq_P$ DOMINATING SET



# VERTEX-COVER $\leq_P$ CLIQUE

- VERTEX-COVER
  – Instance: a graph G and a positive integer k
  – Question: is there a *vertex cover* (i.e. set of vertices "covering" all edges) of size k or less?

- CLIQUE
  – Instance: a graph G and a positive integer p
  – Question: is there a *clique* (i.e. set of vertices all adjacent to each other) of size p or more?

# VERTEX-COVER $\leq_P$ CLIQUE

- Consider a third problem:

  INDEPENDENT SET

  – Instance: a graph $G$ and a positive integer $q$

  – Question: is there an independent set (i.e. set of vertices no-one adjacent to each other) of size $q$ or more?

- For a graph $G=(V,E)$, the following statements are equivalent:

  – $V'$ is a *vertex cover* for $G$

  – $V\setminus V'$ is an *independent set* for $G$

  – $V\setminus V'$ is a *clique* in the complement $G^c$ of $G$

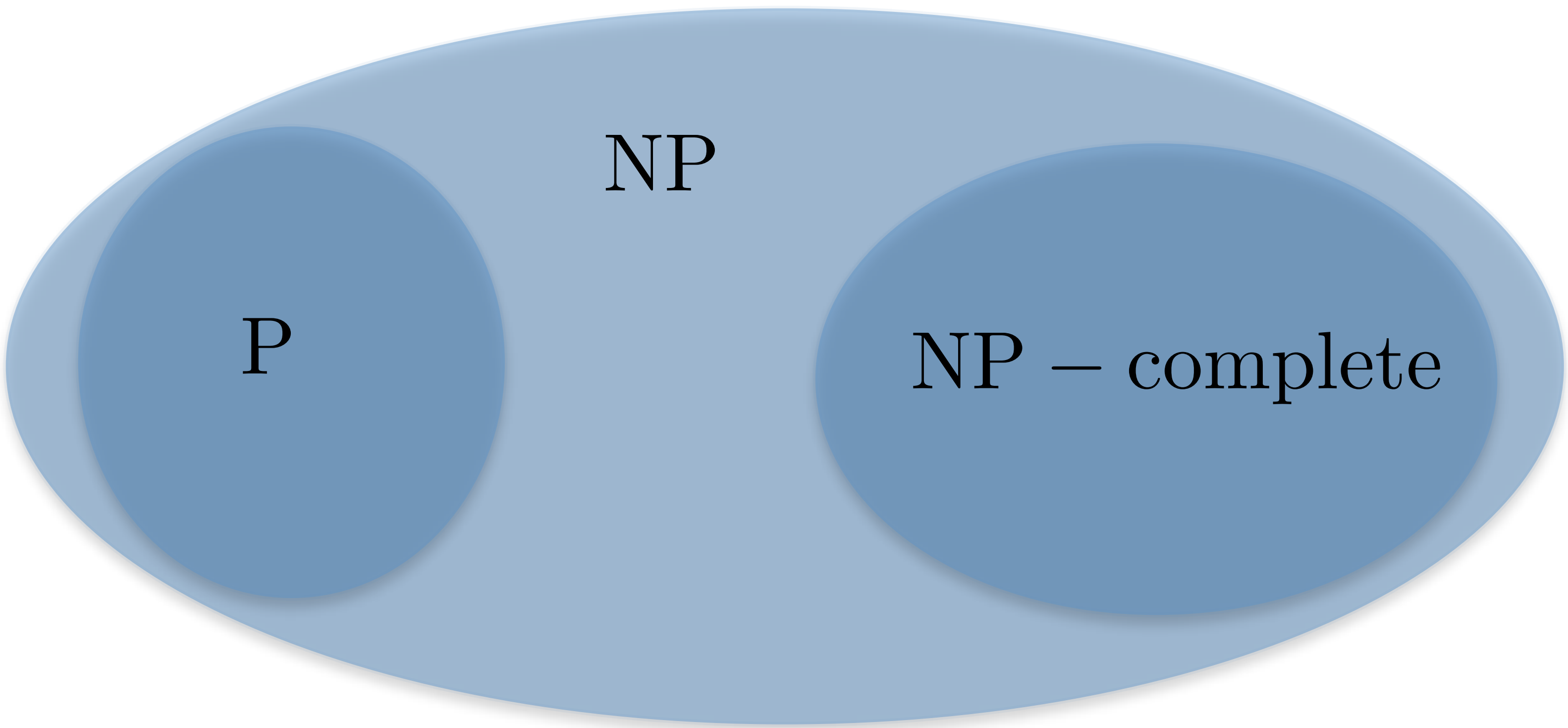
# Reductions - consequences

- Def: $A \leq_P B$: There is a procedure that transforms any instance $I_A$ of A into an instance $I_B$ of B with the following characteristics:
  – the transformation takes polynomial time
  – the answer for $I_A$ is yes iff the answer for $I_B$ is yes
- If B can be solved in polynomial time, and $A \leq_P B$, then A can be solved in polynomial time.
- If A is “hard”, then B should be hard too ....

# The class NP-complete

- A decision problem X is NP-complete if
  - X belongs to NP
  - $A \leq_P X$ for all A in NP
- Theorem[Cook-Karp-Levin]: Vertex-Cover is NP-complete
- Corollary: Dominating Set and Clique are NP-complete, and so are many other problems (Knapsack, Hamiltonian circuit, Longest path problem, etc.)

# One view of various classes ...



# Beyond NP-completeness

- On the negative side, there are decision problems that can be proved *not* to be in NP
  – decidable but not in NP
  – undecidable (ouch !!)

- On the positive side, some “hard” optimization problems can become easier to approximate ... unfortunately not all ...