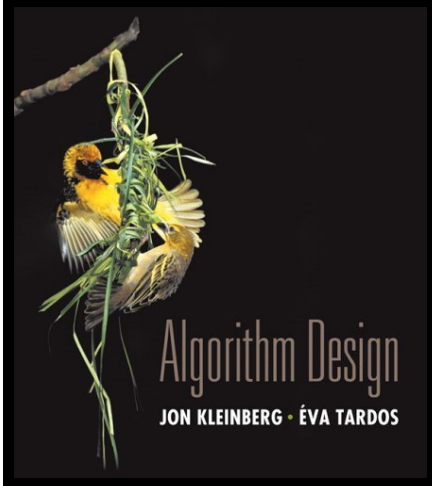


Chapter 8 & 9
Selected Topics

Problem Complexity
P, NP, NP-Complete and PSPACE

Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Algorithm Design Patterns and Anti-Patterns

Algorithm design patterns.	Ex.
▪ Greedy.	$O(n \log n)$ interval scheduling (find maximum subset of mutually compatible jobs).
▪ Divide-and-conquer.	$O(n \log n)$ Closest Pair of Points
▪ Dynamic programming.	$O(n \log n)$ Weighted Interval Scheduling.
▪ Augmenting path.	$O(mn) = O(n^3)$ Bipartite matching.
▪ Reductions.	
▪ Local search.	
▪ Randomization.	
Algorithm design anti-patterns.	
▪ NP-completeness.	$O(n^k)$ algorithm unlikely.
▪ PSPACE-completeness.	$O(n^k)$ certification algorithm unlikely.
▪ Undecidability.	No algorithm possible.

2

Decision Problems

Decision problem.

- X is a set of strings.
- Instance: string s.
- Algorithm A solves problem X: A(s) = **yes** iff $s \in X$.

Polynomial time. Algorithm A runs in poly-time if for every string s, A(s) terminates in at most p(|s|) "steps", where p(·) is a polynomial function.

↑ length of s

PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }
Algorithm. [Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^8$.
Running time $O(\log^{7.5} s)$, for input number s, |s| = logs, in binary representation

Optimization problem.
An optimization problem can be solved with polynomial-time overhead if its decision version can be solved.
Example: what is the largest prime less than n?

3

Definition of P

P. Decision problems for which there is a poly-time algorithm.

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	51, 17	51, 16
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Dynamic programming	niether neither	acgggt ttttta
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix}0&1&1\\2&4&-2\\0&3&15\end{bmatrix}\cdot\begin{bmatrix}4\\2\\36\end{bmatrix}$	$\begin{bmatrix}1&0&0\\1&1&1\\0&1&1\end{bmatrix}\cdot\begin{bmatrix}1\\1\\1\end{bmatrix}$

4

NP

Certification algorithm intuition.

- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. Algorithm C(s, t) is a certifier for problem X if for every string s, $s \in X$ iff there exists a string t such that C(s, t) = `yes`.

"certificate" or "witness"

NP. Decision problems for which there exists a poly-time certifier.

C(s, t) is a poly-time algorithm and $|t| \le p(|s|)$ for some polynomial $p(\cdot)$.

[to determine if a proposed solution t is truly a solution to s or not in poly-time]

Remark. NP stands for non-deterministic polynomial-time. Searching for a proof t that being accepted by C(s, t) is a non-deterministic search over the space of all possible t.

5

Certifiers and Certificates: Composite

COMPOSITES. Given an integer s, is s composite?

Certificate. A nontrivial factor t of s. Note that such a certificate exists iff s is composite. Moreover $|t| \le |s|$.

Certifier.

```
boolean C(s, t) {
   if (t ≤ 1 or t ≥ s)
      return false
   else if (s is a multiple of t)
      return true
   else
      return false
}
```

Instance. s = 437,669.
Certificate. t = 541 or 809. ⟵ 437,669 = 541 × 809

Conclusion. COMPOSITES is in NP.

6

Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ, is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

Ex.

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

certificate t

Conclusion. SAT is in NP.

7

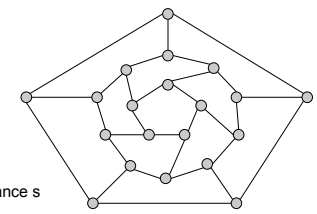
Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

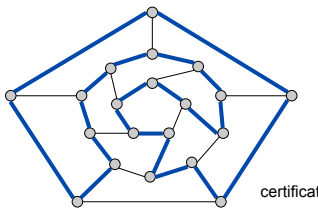
Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Conclusion. HAM-CYCLE is in NP.



instance s



certificate t

8

P, NP, EXP

P. Decision problems for which there is a poly-time algorithm.
EXP. Decision problems for which there is an exponential-time algorithm.
NP. Decision problems for which there is a poly-time certifier.

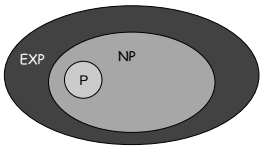
Claim. P ⊆ NP.
Pf. Consider any problem X in P.
- By definition, there exists a poly-time algorithm A(s) that solves X.
- Certificate: t = ε, certifier C(s, t) = A(s). ▪

Claim. NP ⊆ EXP.
Pf. Consider any problem X in NP.
- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with |t| ≤ p(|s|).
- Return `yes`, if C(s, t) returns `yes` for any of these. ▪

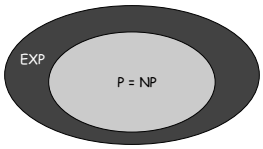
9

The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.



If P ≠ NP



If P = NP

would break RSA cryptography
(and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …
If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

Consensus opinion on P = NP? Probably no.

10

8.1 Polynomial-Time Reductions

Polynomial-Time Reduction

Desiderata'. Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?

don't confuse with reduces from

Reduction. Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

computational model supplemented by special piece of hardware that solves instances of Y in a single step

Notation. $X \leq_P Y$.

Design algorithms. If $X \leq_P Y$ and Y can be solved in polynomial-time, then X can also be solved in polynomial time.

Establish intractability. If $X \leq_P Y$ and X cannot be solved in polynomial-time, then Y cannot be solved in polynomial time.

Establish equivalence. If $X \leq_P Y$ and $Y \leq_P X$, we use notation $X \equiv_P Y$.

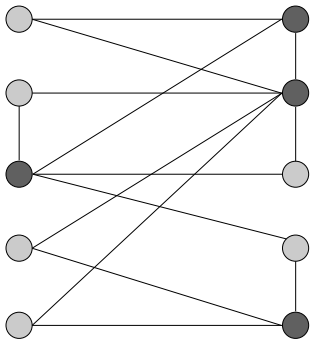
Purpose. Classify problems according to relative difficulty.

13

Independent Set

INDEPENDENT SET: Given a graph G = (V, E) and an integer k, is there a subset of vertices S ⊆ V such that |S| ≥ k, and for each edge at most one of its endpoints is in S?

Ex. Is there an independent set of size ≥ 6? Yes.
Ex. Is there an independent set of size ≥ 7? No.



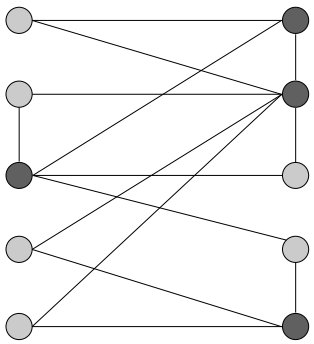
independent set

14

Vertex Cover

VERTEX COVER: Given a graph G = (V, E) and an integer k, is there a subset of vertices S ⊆ V such that |S| ≤ k, and for each edge, at least one of its endpoints is in S?

Ex. Is there a vertex cover of size ≤ 4? Yes.
Ex. Is there a vertex cover of size ≤ 3? No.

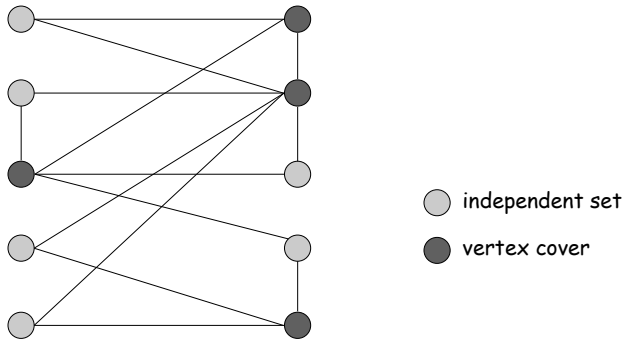


vertex cover

15

Vertex Cover and Independent Set

Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
Pf. We show S is an independent set iff V - S is a vertex cover.



16

Vertex Cover and Independent Set

Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
Pf. We show S is an independent set iff V - S is a vertex cover.

$\Rightarrow$

- Let S be any independent set.
- Consider an arbitrary edge (u, v).
- S independent $\Rightarrow$ $u \notin S$ or $v \notin S$ $\Rightarrow$ $u \in V - S$ or $v \in V - S$.
- Thus, V - S covers (u, v).

$\Leftarrow$

- Let V - S be any vertex cover.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since V - S is a vertex cover.
- Thus, no two nodes in S are joined by an edge $\Rightarrow$ S independent set. ▪

17

8.4 NP-Completeness

NP-Complete

NP-complete. A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$.

Theorem. Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff P = NP.

Pf. $\Leftarrow$ If P = NP then Y can be solved in poly-time since Y is in NP.

Pf. $\Rightarrow$ Suppose Y can be solved in poly-time.

- Let X be any problem in NP. Since $X \leq_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P. /
- We already know P $\subseteq$ NP. Thus P = NP. ▪

NP-hard. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
A decision problem such that every problem in NP reduces to it. not necessarily in NP

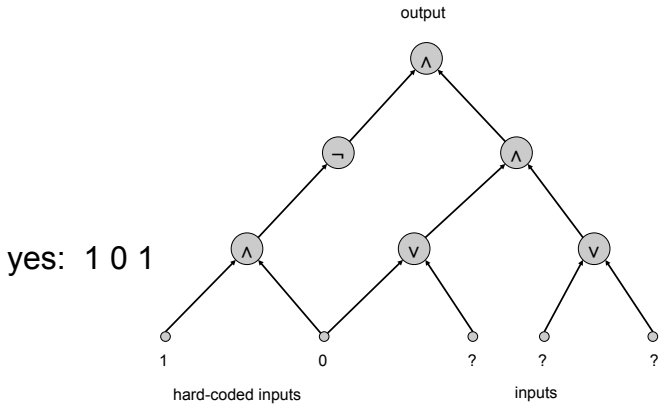
Example: Circuit satisfiability

NP-hard search problem. A problem such that every problem in NP reduces to it. not necessarily a yes/no problem

20

Circuit Satisfiability

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?



Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

21

The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Pf. (sketch)

- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

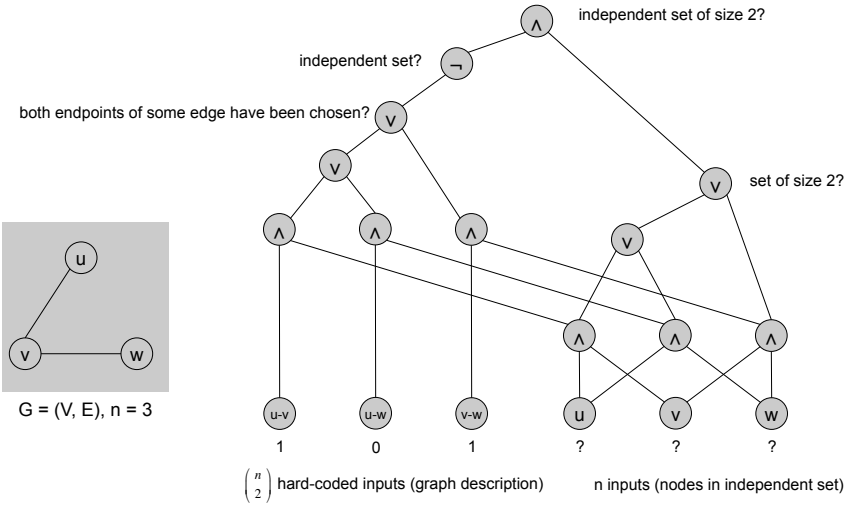
  sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- Consider some problem X in NP. It has a poly-time certifier C(s, t). To determine whether s is in X, need to know if there exists a certificate t of length p(|s|) such that C(s, t) = `yes`.
- View C(s, t) as an algorithm on |s| + p(|s|) bits (input s, certificate t) and convert it into a poly-size circuit K.
  - first |s| bits are hard-coded with s
  - remaining p(|s|) bits represent bits of t
- Circuit K is satisfiable iff C(s, t) = `yes`.

22

Example

Ex. Construction below creates a circuit K whose inputs can be set so that K outputs true iff graph G has an independent set of size 2.



23

Establishing NP-Completeness

Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y.

- Step 1. Show that Y is in NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_P Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \leq_P X \leq_P Y$.

- By transitivity, $W \leq_P Y$.
- Hence Y is NP-complete. ▪

($W \leq_P X$: by definition of NP-complete; $X \leq_P Y$: by assumption)

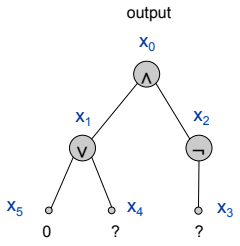
24

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.

Pf. Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

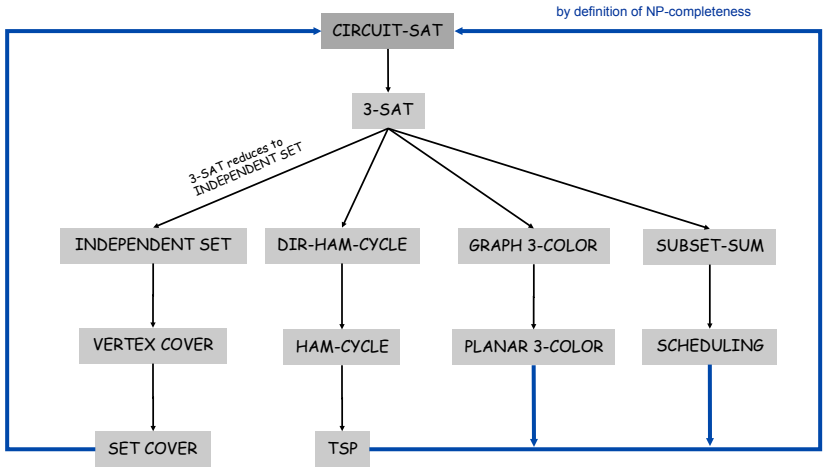
- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3$ $\Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5$ $\Rightarrow$ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2$ $\Rightarrow$ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$
- Hard-coded input values and output value.
  - $x_5 = 0$ $\Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ $\Rightarrow$ add 1 clause: x_0
- Final step: turn clauses of length < 3 into clauses of length exactly 3. ▪



25

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!



26

Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.

Factoring: Given two integers x and y, does x have a nontrivial factor less than y?

27

PSPACE

P. Decision problems solvable in polynomial time.

PSPACE. Decision problems solvable in polynomial space.

Observation. P $\subseteq$ PSPACE.

↑

poly-time algorithm can consume only polynomial space

Claim. 3-SAT is in PSPACE.

Pf.

- Enumerate all 2^n possible truth assignments using counter.
- Check each assignment to see if it satisfies all clauses. ▪

Theorem. NP $\subseteq$ PSPACE.

Pf. Consider arbitrary problem Y in NP.

- Since Y $\leq_P$ 3-SAT, there exists algorithm that solves Y in poly-time plus polynomial number of calls to 3-SAT black box.
- Can implement black box in poly-space. ▪

28

PSPACE-Complete

PSPACE. Decision problems solvable in polynomial space.

PSPACE-Complete. Problem Y is PSPACE-complete if
(i) Y is in PSPACE and
(ii) for every problem X in PSPACE, $X \leq_P Y$.

Theorem. [Stockmeyer-Meyer 1973] QSAT is PSPACE-complete.
Quantified 3-SAT $\exists x_1 \forall x_2 \cdots \exists x_{n-2} \forall x_{n-1} \exists x_n \Phi(x_1, \ldots, x_n)?$

Theorem. PSPACE $\subseteq$ EXPTIME.
Pf. Previous algorithm solves QSAT in exponential time, and QSAT is PSPACE-complete. ▪

Summary. P $\subsetneq$ NP $\subsetneq$ PSPACE $\subsetneq$ EXPTIME.
it is known that P $\neq$ EXPTIME, but unknown which inclusion is strict; conjectured that all are

29

Quantified Satisfiability

QSAT. Let $\Phi(x_1, \ldots, x_n)$ be a Boolean CNF formula. Is the following propositional formula true?

$$\exists x_1 \; \forall x_2 \; \exists x_3 \; \forall x_4 \ldots \; \forall x_{n-1} \; \exists x_n \; \Phi(x_1, \ldots, x_n)$$

↑
assume n is odd

Intuition. Amy picks truth value for x_1, then Bob for x_2, then Amy for x_3, and so on. Can Amy satisfy Φ no matter what Bob does?

Ex. $(x_1 \vee x_2) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$
Yes. Amy sets x_1 true; Bob sets x_2; Amy sets x_3 to be same as x_2.

Ex. $(x_1 \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$
No. If Amy sets x_1 false; Bob sets x_2 false; Amy loses;
if Amy sets x_1 true; Bob sets x_2 true; Amy loses.

30

PSPACE-Complete Problems

A number of basic problems in AI are PSPACE-complete:
Planning, Game, Quantification (QSAT)

More PSPACE-complete problems.

- Competitive facility location.
- Natural generalizations of games.
  - Othello, Hex, Geography, Rush-Hour, Instant Insanity
  - Shanghai, go-moku, Sokoban
- Given a memory restricted Turing machine, does it terminate in at most k steps?
- Do two regular expressions describe different languages?
- Is it possible to move and rotate complicated object with attachments through an irregularly shaped corridor?
- Is a deadlock state possible within a system of communicating processors?

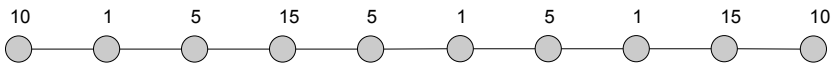
31

Competitive Facility Location

COMPETITIVE-FACILITY is PSPACE-complete.

Input. Graph with positive edge weights, and target B.
Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors has been selected.

Competitive facility location. Can second player guarantee at least B units of profit?



Yes if B = 20; no if B = 25.

32