

Algorithm Design and Analysis

CSE
565

LECTURE 37
Computational Intractability

- Classes P and NP
- NP-completeness

Sofya Raskhodnikova

11/28/2007

S. Raskhodnikova; based on slides by K. Wayne

CSE
565

Review Question

FACTORING: factor a given integer
Question: Give a related decision problem.

Reminder:

Decision problem: X is a set of strings.

Instance: string s .

Algorithm A solves problem X : $A(s) = \text{yes}$ iff $s \in X$.

Answer:

PRIMES: $X = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \dots \}$

11/28/2007

S. Raskhodnikova; based on slides by K. Wayne

Polynomial Time

Algorithm A runs in polynomial time if for every input string s , $A(s)$ terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial.

↑
length of s

PRIMES has a polynomial time algorithm.
[Agrawal-Kayal-Saxena, 2002]
 $p(|s|) = |s|^3$.

Definition of Class 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 ttttaa
LSOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & 2 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

Definition of Class 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 $C(s, t) = \text{yes}$.

↑
"certificate" or "witness"

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

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

Remark. NP stands for **nondeterministic** polynomial-time.

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 $|t| \leq |s|$.

Certifier.

```
boolean CompositesCertifier(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 . $\leftarrow 437,669 = 541 \times 809$

Conclusion. COMPOSITES is in NP.

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

instance s

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

certificate t

Conclusion. SAT is in NP.

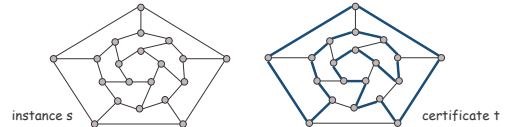
Certifiers and Certificates: Hamiltonian Cycle

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.



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, i.e., an algorithm that runs in time $O(2^{p(|s|)})$ for some polynomial $p(\cdot)$.

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

Claim. $P \subseteq NP$.

Pf. Consider any problem X in P.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X .
- Certificate: $t = \epsilon$, certifier $C(s, t) = A(s)$. •

Claim. $NP \subseteq EXP$.

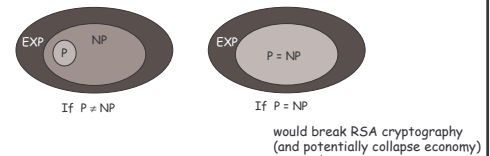
Pf. Consider any problem X in NP.

- 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 $|t| \leq p(|s|)$.
- Return *yes*, if $C(s, t)$ returns *yes* for any of these. •

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 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.

NP-Complete

NP-complete. A problem Y in NP-complete if Y is in NP

$X \leq_{p, \text{Karp}} Y$ for every problem X in NP.

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.

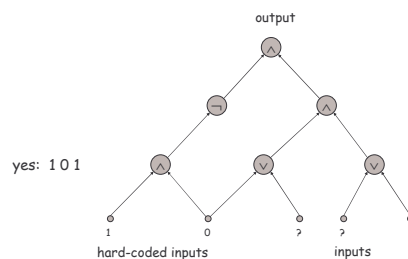
Pf. \Rightarrow Suppose Y can be solved in poly-time.

- Let X be any problem in NP. Since $X \leq_{p, \text{Karp}} Y$, we can solve X in poly-time. This implies $NP \subseteq P$.
- We already know $P \subseteq NP$. Thus $P = NP$. •

Fundamental question. Do there exist "natural" NP-complete problems?

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?



The "First" NP-Complete Problem

Theorem. **CIRCUIT-SAT** is NP-complete. [Cook 1971, Levin 1973]
 Pf. (sketch)

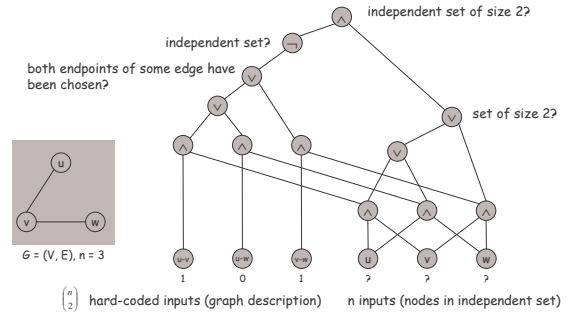
- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

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 $C(s, t) = \text{yes}$.
- 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 $C(s, t) = \text{yes}$.

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.



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 $X \leq_{p, \text{Karp}} Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_{p, \text{Karp}} Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \leq_{p, \text{Karp}} X \leq_{p, \text{Karp}} Y$.

- By transitivity, $W \leq_{p, \text{Karp}} Y$.
- Hence Y is NP-complete.

3-SAT is NP-Complete

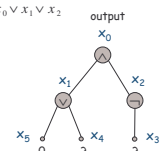
Theorem. **3-SAT** is NP-complete.

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: $\overline{x_2 \vee x_3}, x_2 \vee \overline{x_3}$
 - $x_1 = x_4 \vee x_5 \Rightarrow$ add 3 clauses: $\overline{x_1 \vee \overline{x_4}}, \overline{x_1 \vee \overline{x_5}}, x_1 \vee x_4 \vee x_5$
 - $x_0 = x_1 \wedge x_2 \Rightarrow$ add 3 clauses: $\overline{x_0 \vee \overline{x_1}}, \overline{x_0 \vee \overline{x_2}}, x_0 \vee x_1 \vee x_2$

Hard-coded input values and output value.

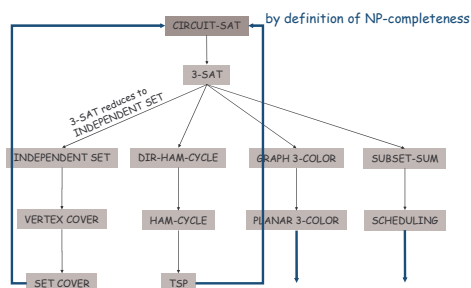
- $x_5 = 0 \Rightarrow$ add 1 clause: $\overline{x_5}$
- $x_0 = 1 \Rightarrow$ add 1 clause: x_0



- Final step: turn clauses of length < 3 into clauses of length exactly 3 by adding extra variables: $x_1 \vee x_2$ is satisfiable iff $(x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \overline{y})$ is satisfiable.

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce (with Karp reductions) to one another!



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, Traveling Salesman.
- 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. For most search problems, if the corresponding decision problem is in P, the search problem can be solved in polynomial time.

Notable exceptions. Decision problem: Graph isomorphism. Search problems: Factoring, Nash equilibrium.

Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]

- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
 - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.

- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: Istrail proves 3D problem NP-complete.

More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.
Biology: protein folding.

Chemical engineering: heat exchanger network synthesis.

Civil engineering: equilibrium of urban traffic flow.

Economics: computation of arbitrage in financial markets with friction.

Electrical engineering: VLSI layout.

Environmental engineering: optimal placement of contaminant sensors.

Financial engineering: find minimum risk portfolio of given return.

Game theory: find Nash equilibrium that maximizes social welfare.

Genomics: phylogeny reconstruction.

Mechanical engineering: structure of turbulence in sheared flows.

Medicine: reconstructing 3-D shape from biplane angiogram.

Operations research: optimal resource allocation.

Physics: partition function of 3-D Ising model in statistical mechanics.

Politics: Shapley-Shubik voting power.

Pop culture: Minesweeper consistency.

Statistics: optimal experimental design.