COMP 355
Advanced Algorithms

NP-Completeness: General Definitions
Chapter 8 Intro (KT)

Efficiency and Polynomial Time

The notion of what we mean by efficient is quite vague.

- If n is small, 2^n run time may be just fine
- When n is huge, even n^2 may be unacceptably slow

2 general classes of combinatorial problems

- Those involving *brute-force search* of all feasible solutions, whose worst-case running time is an *exponential function* of the input size
- Those that are based on a *systematic solution*, whose worst-case running time is a *polynomial function* of the input size.

An algorithm is said to run in *polynomial time* if its worst-case running time is $\boldsymbol{O(n^c)}$, where c is a nonnegative constant.

Higher worst-case running times, such as c^n, $n!$, and n^n are not polynomial time.

You can't be serious!

Arguments against using polynomial-time as the definition of efficiency

- When n is small, run time of 2^n much faster than $O(n^{20})$
- Many problems for which good average case solutions exist, but the worst case complexity may be very bad
- On modern architectures, practical efficiency is a function of many issues that have to do with the machine's internal architecture

Mathematical advantages of defining "efficiently solvable" to be "worst-case polynomial time solvable"

- The composition of two polynomials is a polynomial
  - Example: an algorithm that makes $O(n^2)$ calls to a function that takes $O(n^3)$ time runs in $O(n^5)$ time, which is still polynomial.
- No need to worry about the distribution of inputs

When n is sufficiently large, exponential time algorithms (2^n time) are not efficient

The Emergence of Hard Problems

"hard" problem = no known efficient algorithmic solutions exist for these problems.

Hard problems (NP-complete)	Easy problems (in P)
3SAT	2SAT
Traveling Salesman Problem (TSP)	Minimum Spanning Tree (MST)
Longest (Simple) Path	Shortest Path
3D Matching	Bipartite Matching
Knapsack	Unary Knapsack
Independent Set in Graphs	Independent Set in Trees
Integer Linear Programming	Linear Programming
Hamiltonian Cycle	Eulerian Cycle
Balanced Cut	Minimum Cut

Reasonable Input Encodings

- Must define terms precisely
- Treat the input to our problems as a string over some alphabet that has a constant number, but at least two, characters (e.g., a binary bit string or a Unicode encoding).
- All the representations we have seen this semester (e.g., sets as lists, graphs as adjacency lists or adjacency matrices, etc.) are considered to be reasonable.
- To determine whether some new representation is reasonable:
  - It should be as concise as possible (in the worst case)
  - It should be possible to convert from an existing reasonable representation to this new form in polynomial time.

Decision Problems and Languages

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 some polynomial.

Decision Problem Example

MST Decision Problem: "Given a weighted graph G and an integer z, does G have a spanning tree whose weight is at most z?

Language: a set (finite or infinite) of strings.

Formulating MST decision problem as a language recognition problem.

- Define a language MST encoding the minimum spanning tree problem as:

  MST = {(G, z) | G has a minimum spanning tree of weight at most z}.

What does it mean to solve the decision problem?

- When presented with a specific input string x = serialize(G, z), the algorithm would answer "yes" if x ∈ MST (G has a spanning tree of weight at most z) and "no" otherwise.
- If "yes", algorithm *accepts* the input and otherwise it *rejects* the input.
- Decision problems are equivalent to language membership problems.

The Class P

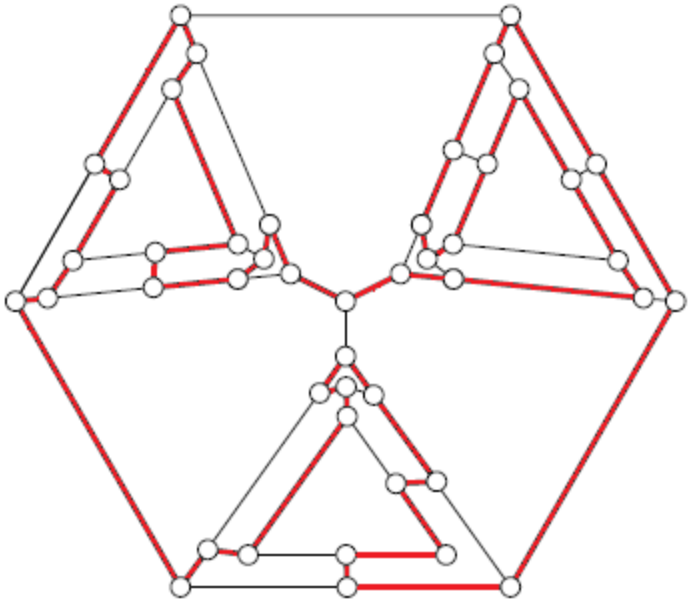
Definition: P is the set of all languages (i.e., decision problems) for which membership can be determined in (worst case) polynomial time.

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

Not all Decision Problems are in P

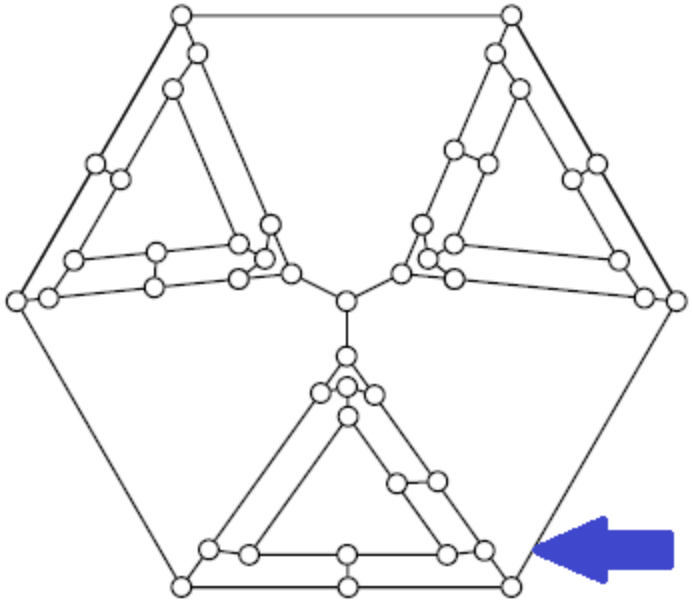
HC = {G | G has a simple cycle that visits every vertex of G}.

Hamiltonian



(a)

Hamiltonian?



(b)

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`.

NP. Set of all languages that can be verified in polynomial time.

Remark. NP stands for nondeterministic polynomial-time.

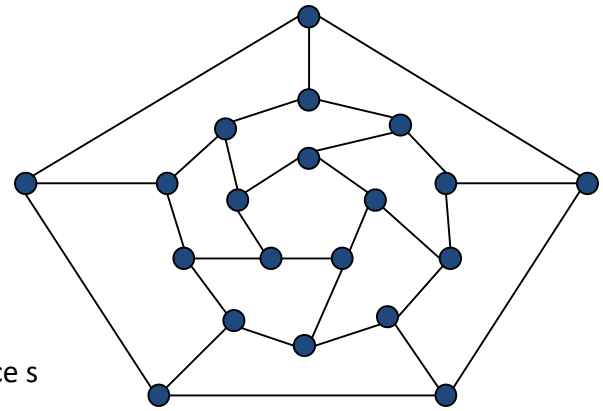
Certifiers and Certificates: Hamiltonian Cycle

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

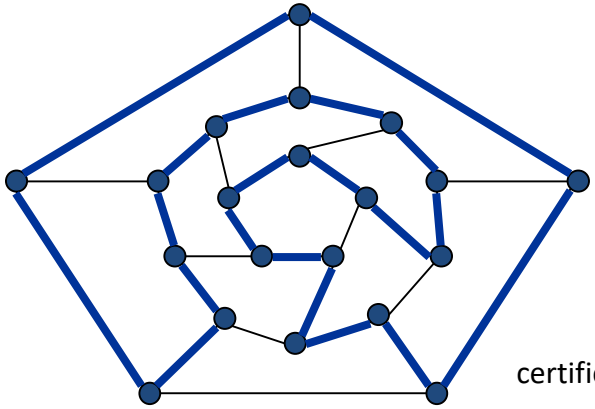
Certificate. A permutation of the n nodes.

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

Certifiers and Certificates: 3-Satisfiability

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

Certificate. An assignment of truth values to the k 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.

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. $P \subseteq NP$.

Pf. Consider any problem X in P.

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

Claim. $NP \subseteq EXP$.

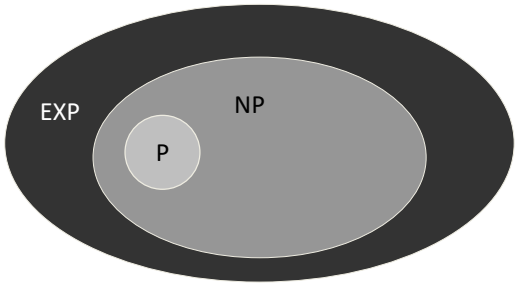
Pf. Consider any problem X in NP.

- 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. ▪

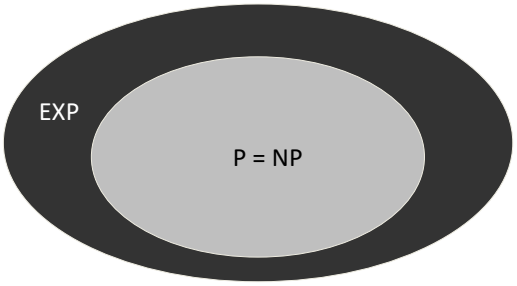
The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.



If P ≠ NP



If P = NP

would break RSA cryptography (and potentially collapse economy)

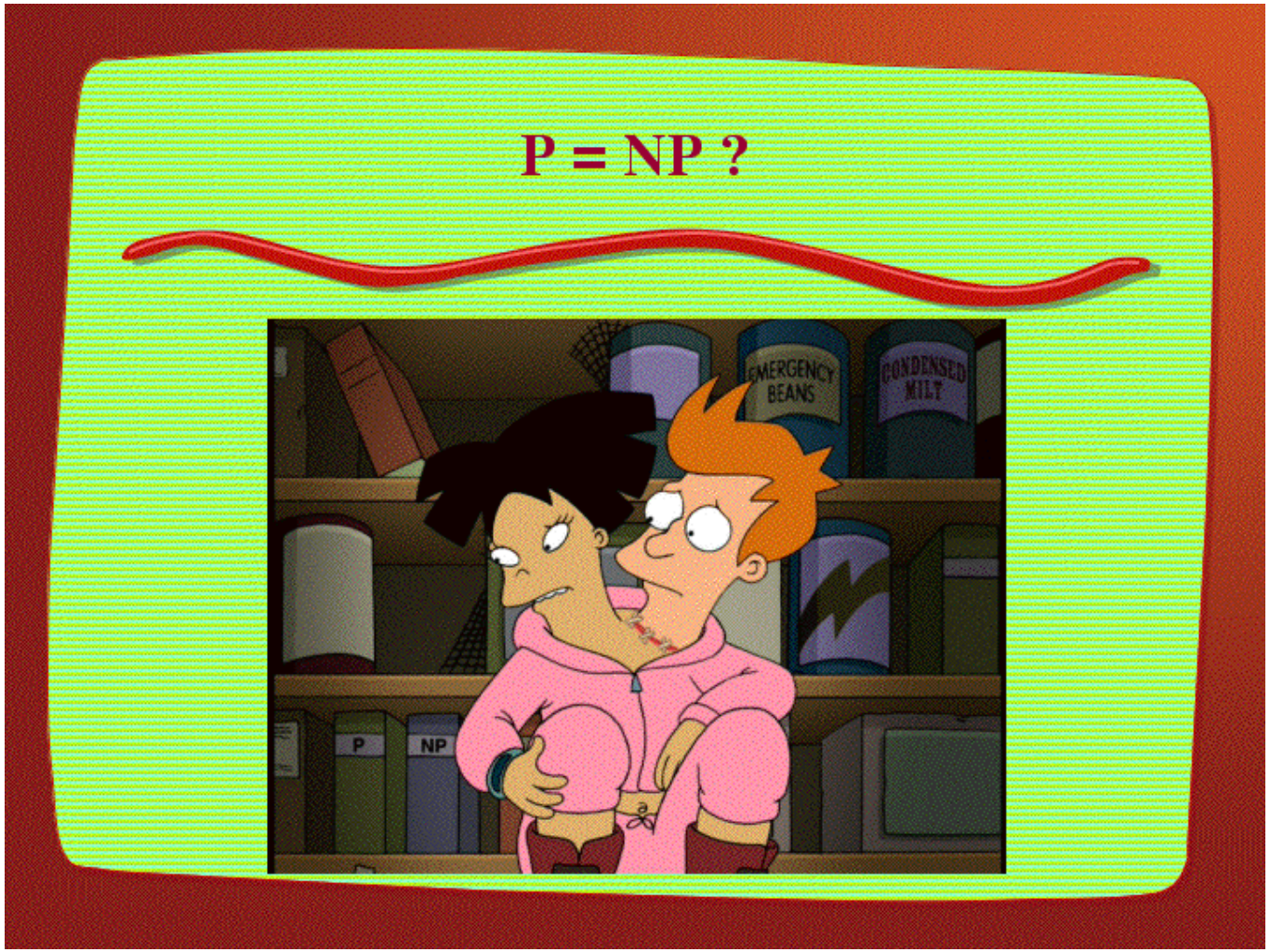
If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …

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

Consensus opinion on P = NP? Probably no.

The Simpson's: P = NP?

Futurama: P = NP?



Looking for a Job?

Some writers for the Simpsons and Futurama.

- J. Steward Burns. M.S. in mathematics, Berkeley, 1993.
- David X. Cohen. M.S. in computer science, Berkeley, 1992.
- Al Jean. B.S. in mathematics, Harvard, 1981.
- Ken Keeler. Ph.D. in applied mathematics, Harvard, 1990.
- Jeff Westbrook. Ph.D. in computer science, Princeton, 1989.