Algorithm Complexity: P vs NP

(5) P (deterministic polynomial time) ... aka PTIME

- problems solvable via deterministic algorithms (TMs) in time $O(n^{O(1)})$
- these problems often considered “fast”, “tractible”, “feasible”, “efficient”
- note that according to this definition $O(n^{100})$ is “efficient”

(6) (PTIME) Reductions

- One problem is solved by mapping it to another problem ...

From Cormen et al., “Introduction to Algorithms”, 3rd Ed.
- often written as $A \leq_P B$, which says $A$ is polynomial-time reducible to $B$
Algorithm Complexity: P vs NP

(7) NP (non-deterministic polynomial time)
• problems solvable via non-deterministic algorithms (TMs) in time $O(n^{O(1)})$
• note that determinism and non-determinism have different assumptions ...

Basic Idea:
• with non-determinism, solutions (certificates) “guessed” and then checked
• where guessing certificates is assumed to be fast (at most polynomial time)
• and certificates can be verified (checked) in polynomial time
• implies $P \subseteq NP$ ... since $P$ implies answers found in polynomial time

P vs NP:
• P consists of problems that can be solved quickly
• NP consists of problems with solutions that can be verified quickly

Open Problem: Is $P \subsetneq NP$ or is $P = NP$?
• there are problems in NP with no known PTIME solutions!
• suggesting $P \subsetneq NP$ ...
• However, if PTIME solutions found, then $P = NP$
• $P \not= NP$ is one of the 7 Millenium Prize problems ($1,000,000 award!$)

General consensus is $P \subsetneq NP$ ...
• a PTIME algo. for any of the “hardest” NP problems would prove $P = NP$
• however, it has been 50+ years with no such PTIME solutions
Algorithm Complexity: P vs NP

(8) **NP-Hard Problems**
  - a problem where all problems in NP can be reduced to it
  - problems as hard and harder than NP (e.g., EXPTIME verification)

(9) **NP-Complete Problems**
  - problems that are both NP-hard and are in NP
  - these are the “hardest” problems in NP
  - considered to be “hard”, “intractible”, “infeasible”

*Note:* Also possibly NP-Intermediate problems, between P and NP-Complete

Some classic NP-Complete problems
- Boolean Formula Satisfiability (SAT) ... \( \land, \lor, \text{ and } \neg \) over Boolean vars
- CNFSAT ... satisfiability of conjunctive normal form Boolean formulas
- 3-SAT ... at most 3 literals per clause (e.g., \((x_i \lor x_j \neg \bar{x}_k) \land \ldots\))
- Traveling Salesman Problem (TSP) ... find “tour” in weighted graph
- SUBSET-SUM (SSP) ... does a sub-collection sum to a given value
- Knapsack, Clique, Independent Set, Graph Coloring, Hamiltonian Cycle
- plus many more
Algorithm Complexity: P vs NP

Every NP-complete problem is as hard as the others (via reductions)

- thus, showing one has a PTIME solution means all do
- NP-complete problems are thus “equivalent” w.r.t. PTIME reductions

(10) **To prove a problem is NP-Hard use a reduction**

- i.e., reduce an NP-Complete problem to it
- for example, show \( \text{SAT} \leq_p \text{A} \) ... implies \( \text{A} \) is at least as hard as SAT

(11) **To prove a problem is NP-Complete ...**

- show it is NP Hard (see 10)
- show it is in NP ...
  - via a PTIME (certificate) verification algorithm
  - or by reducing it to an NP-Complete problem (e.g., show that \( \text{A} \leq_p \text{SAT} \))