P, *NP*, and *NP*-complete Problems

DEFINITION 1 We say that an algorithm solves a problem in polynomial time if its worst-case time efficiency belongs to $O(p(n))$ where $p(n)$ is a polynomial of the problem's input size n. (Note that since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.) Problems that can be solved in polynomial time are called ***tractable***, problems that cannot be solved in polynomial time are called ***intractable***.

DEFINITION 2 Class P is a class of decision problems that can be solved in polynomial time by (deterministic) algorithms. This class of problems is called *polynomial*.

Hamiltonian circuit Determine whether a given graph has a Hamiltonian circuit (a path that starts and ends at the same vertex and passes through all the other vertices exactly once).

Traveling salesman Find the shortest tour through n cities with known positive integer distances between them (find the shortest Hamiltonian circuit in a complete graph with positive integer weights).

Knapsack problem Find the most valuable subset of n items of given positive integer weights and values that fit into a knapsack of a given positive integer capacity.

Partition problem Given n positive integers, determine whether it is possible to partition them into two disjoint subsets with the same sum.

Bin packing Given n items whose sizes are positive rational numbers not larger than 1, put them into the smallest number of bins of size 1.

Graph coloring For a given graph, find its chromatic number (the smallest number of colors that need to be assigned to the graph's vertices so that no two adjacent vertices are assigned the same color).

Integer linear programming Find the maximum (or minimum) value of a linear function of several integer-valued variables subject to a finite set of constraints in the form of linear equalities and/or inequalities.

DEFINITION 3 A ***nondeterministic algorithm*** is a two-stage procedure that takes as its input an instance I of a decision problem and does the following.

Nondeterministic ("guessing") stage: An arbitrary string S is generated that can be thought of as a candidate solution to the given instance I (but may be complete gibberish as well).

Deterministic ("verification") stage: A deterministic algorithm takes both I and S as its input and outputs yes if S represents a solution to instance I. (If S is not a solution to instance I, the algorithm either returns no or is allowed not to halt at all.)

DEFINITION 4 Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms. This class of problems is called ***nondeterministic polynomial***.

Most decision problems are in NP. First of all, this class includes all the problems in P:

$$P \subseteq NP.$$

NP-Complete Problems

Informally, an *NP*-complete problem is a problem in *NP* that is as difficult as any other problem in this class because, by definition, any other problem in *NP* can be reduced to it in polynomial time (shown symbolically in Figure 11.6).

Here are more formal definitions of these concepts.

DEFINITION 5 A decision problem D_1 is said to be ***polynomially reducible*** to a decision problem D_2 if there exists a function t that transforms instances of D_1 to instances of D_2 such that

1. t maps all yes instances of D_1 to yes instances of D_2 and all no instances of D_1 to no instances of D_2;
2. t is computable by a polynomial-time algorithm.

DEFINITION 6 A decision problem D is said to be ***NP-complete*** if

1. it belongs to class *NP*;
2. every problem in *NP* is polynomially reducible to D.

The optimization versions of combinatorial problems (traveling salesman problem and the knapsack problem) fall in the class of ***NP-hard*** *problems-problems* that are at least as hard as NP-complete problems. Hence, there are no known polynomial-time algorithms for these problems, and there are serious theoretical reasons to believe that such algorithms do not exist.

Cook's theorem

- In computational complexity theory, the **Cook–Levin theorem**, also known as **Cook's theorem**, states that the Boolean satisfiability problem is NP-complete. That is, any problem in NP can be reduced in polynomial time by a deterministic Turing machine to the problem of determining whether a Boolean formula is satisfiable.
- An important consequence of the theorem is that if there exists a deterministic polynomial time algorithm for solving Boolean satisfiability, then there exists a deterministic polynomial time algorithm for solving *all* problems in NP. Crucially, the same follows for any NP complete problem.
- The question of whether such an algorithm exists is called the P versus NP problem and it is widely considered the most important unsolved problem in theoretical computer science. The theorem is named after Stephen Cook and Leonid Levin.