NP-complete problems
- Informally, these are the hardest problems in the class NP
- If any NP-complete problem can be solved by a polynomial time deterministic algorithm, then every problem in NP can be solved by a polynomial time deterministic algorithm
- But no polynomial time deterministic algorithm is known to solve any of them

Examples of NP-complete problems
- Traveling salesman problem
- Hamiltonian cycle problem
- Clique problem
- Subset sum problem
- Boolean satisfiability problem
- Many thousands of other important computational problems in computer science, mathematics, economics, manufacturing, communications, etc.

Polynomial-time reduction
Let $L_1$ and $L_2$ be two languages over alphabets $\Sigma_1$ and $\Sigma_2$, respectively. $L_1$ is said to be polynomial-time reducible to $L_2$ if there is a total function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ for which
1) $x \in L_1$ if and only if $f(x) \in L_2$, and
2) $f$ can be computed in polynomial time
The function $f$ is called a polynomial-time reduction.

Another view
One decision problem is polynomial-time reducible to another if a polynomial time algorithm can be developed that changes each instance of the first problem to an instance of the second such that a yes (or no) answer to the second problem entails a yes (or no) answer to the first.

Exercise
In English, describe a Turing machine that reduces $L_1$ to $L_2$ in polynomial time. Using big-O notation, give the time complexity of the machine that computes the reduction.
- $L_1 = \{a^ib^ia^i\}$  $L_2 = \{c^jd^j\}$
- $L_1 = \{a^i(bb)^i\}$  $L_2 = \{a^ib^i\}$
A more interesting polynomial-time reduction

- The Hamiltonian cycle problem can be polynomial-time reduced to the traveling salesman problem.
- For any undirected graph $G$, we show how to construct an undirected weighted graph $G'$ and a bound $B$ such that $G$ has a Hamiltonian cycle if and only if there is a tour in $G'$ with total weight bounded by $B$.
- Given $G = (V,E)$, let $B = 0$ and define $G' = (V,E')$ as the complete graph with the following weights assigned to edges:
  \[ w_{i,j} = \begin{cases} 
  0 & \text{if } (i,j) \in E \\
  1 & \text{if } (i,j) \not\in E 
  \end{cases} \]
- $G$ has a Hamiltonian cycle if and only if $G'$ has a tour with total weight 0.

Exercise

- **Partition Problem**: Given a finite set of integers, determine if it can be partitioned into two sets such that the sum of all integers in the first set equals the sum of all integers in the second set.
- **Subset Sum Problem**: Given a set of integers and a number $t$, determine if there is a subset of these integers whose sum is $t$.
- Show that the Partition Problem is polynomial-time reducible to the Subset Sum Problem.

Examples of problem reductions

- **SAT**: Is there a satisfying assignment for a proposition in conjunctive normal form?
- **3-SAT**: Same as above except every clause in proposition has exactly three literals.
- **HAM-CYCLE**: Given an undirected graph, determine whether it contains a Hamiltonian cycle (a path that starts at one node, visits every other node exactly once, and returns to start).
- **TSP**: Given a fully-connected weighted graph, find a least-weight tour of all nodes (cities).

SAT (Boolean satisfiability)

- In order to use polynomial-time reductions to show that problems are NP-complete, we must be able to directly show that at least one problem is NP-complete, without using a polynomial-time reduction.
- Cook proved that the Boolean satisfiability problem (denoted SAT) is NP-complete. He did not use a polynomial-time reduction to prove this.
- This was the first problem proved to be NP-complete.

Strategy for proving a problem is NP-complete

- Show that it belongs to the class NP by describing a nondeterministic Turing machine that solves it in polynomial time. (This establishes an upper bound on the complexity of the problem.)
- **Exercise**: In English, describe a nondeterministic algorithm that solves the satisfiability problem.
- Show that the problem is NP-hard by showing that another NP-hard problem is polynomial-time reducible to it. (This establishes a lower bound on the complexity of the problem.)

Definition of NP-Complete

- A problem is NP-Complete if
  1. It is an element of the class NP
  2. Another NP-complete problem is polynomial-time reducible to it
- A problem that satisfies property 2, but not necessarily property 1, is NP-hard.
P ≠ NP?

- **Theorem**: If any NP-complete problem can be solved by a polynomial-time deterministic algorithm, then P = NP. If any problem in NP cannot be solved by a polynomial-time deterministic algorithm, then NP-complete problems are not in P.
- This theorem makes NP-complete problems the focus of the P=NP question.
- Most theoretical computer scientists believe that P ≠ NP. But no one has proved this yet.

What should we do?

- Just because a problem is NP-complete, doesn’t mean we should give up on trying to solve it.
- For some NP-complete problems, it is possible to develop algorithms that have average-case polynomial complexity (despite having worst-case exponential complexity)
- For other NP-complete problems, approximate solutions can be found in polynomial time. Developing good approximation algorithms is an important area of research.

Complexity classes

- A complexity class is a class of problems grouped together according to their time and/or space complexity
- NC: can be solved very efficiently in parallel
- P: solvable by a DTM in poly-time (can be solved efficiently by a sequential computer)
- NP: solvable by a NTM in poly-time (a solution can be checked efficiently by a sequential computer)
- PSPACE: solvable by a DTM in poly-space
- NPSPACE: solvable by a NTM in poly-space
- EXPTIME: solvable by a DTM in exponential time

Relationships between complexity classes

- NC ⊆ P ⊆ NP ⊆ PSPACE = NPSPACE ⊆ EXPTIME
- P ≠ EXPTIME
- Saying a problem is in NP (P, PSPACE, etc.) gives an upper bound on its difficulty
- Saying a problem is NP-hard (P-hard, PSPACE-hard, etc.) gives a lower bound on its difficulty. It means it is at least as hard to solve as any other problem in NP.
- Saying a problem is NP-complete (P-complete, PSPACE-complete, etc.) means that we have matching upper and lower bounds on its complexity