P, NP, NP-Complete, NP-Hard

Introduction

- All the algorithms studied so far have been polynomial-time algorithms on inputs of size n.
  - The worst-case running time is $O(n^k)$ for some constant k.

- That mean all problems can be solved in polynomial time. **NO**.

Introduction

- There are problems that
  - Can be solved, but not in time $O(n^k)$ for any constant k.
  - Cannot be solved by any computer. (Turing's famous "Halting Problem")
- Problems solvable by polynomial-time algorithms are tractable, or easy.
- Problems that require superpolynomial time are intractable, or hard.

Decision vs. Optimization Problems

- Many problems of interest are optimization problems.
  - Each feasible solution has an associated value, and objective is to find a feasible solution with the best value.
  - Example, a SHORTEST-PATH problem states that given an undirected graph G and vertices u and v, find a path from u to v with the fewest edges.
- Npcompleteness is not directly applicable to such problems.

Contd…

- NP completeness applies directly and also confined to decision problems
  - The answer is simply "yes (1)" or "no (0)".
- An optimization problem can be casted as a related decision problem by imposing a bound on the value to be optimized.
  - For example, a decision problem related to SHORTEST-PATH is PATH: given a directed graph G, vertices u and v, and an integer k, does a path exist from u to v consisting of at most k edges?

Contd…

- If an optimization problem is easy, its related decision problem is easy as well.
- Talking in relevance to NP-completeness, if we can provide evidence that a decision problem is hard, we also provide evidence that its related optimization problem is hard.
- Thus, even though it restricts attention to decision problems, the theory of NP-completeness often has implications for optimization problems as well.

The Class P

- The complexity class P is the set of all decision problems that can be solved with worst-case polynomial time-complexity.
- More specifically, they are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem.
- P is just the set of tractable decision problems.

Example

- Determining if a graph can be colored with 2 colors, i.e. determining whether or not the graph is bipartite?

- Find the shortest path from a single source in a directed graph.

The Class NP

- **NP is not the same as non-polynomial complexity/running time. NP does not stand for not polynomial.**
- **NP = Non-Deterministic polynomial time**
- NP means verifiable in polynomial time
- Verifiable?
  - If we are somehow given a ‘certificate’ of a solution we can verify the legitimacy (correct or incorrect) in polynomial time in the size of the input to the problem.
- All P problems are in NP, i.e. $P \subseteq NP$

Example

- Hamiltonian cycle problem: A cycle passing through all the vertices of a graph is called a **Hamiltonian cycle**.
  - Given a directed graph G = (V,E), a certificate would be a sequence $\langle v_1, v_2, v_3, v_{|V|} \rangle$ of $|V|$ vertices.
  - It can easily be checked in polynomial time that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, 3, \ldots |V| - 1$ and that $(v_{|V|}, v_1) \in E$ as well.
- As another example, for 3-CNF satisfiability, a certificate would be an assignment of values to variables. We could check in polynomial time that this assignment satisfies the boolean formula.

P is a subset of NP

- Since it takes polynomial time to run the program, just run the program and get a solution.
- But is P a proper subset of NP?
  - AN OPEN QUESTION…
- No one knows if P = NP or not

NP - hard

- What are the hardest problems in NP?

$$L_1 \leq_p L_2$$

- That notation means that L_1 is reducible in polynomial time to L_2 .
- The less than symbol basically means that the time taken to solve L_1 is not more than a polynomial factor harder than L_2.

The Class NP-hard

- A problem is said to NP-hard if every problem in NP can be reduced to it in polynomial time .

$$L' \leq_p L \text{ for every } L' \in NP$$

Example

- The halting problem.
- This is the problem that given a program P and input I, will it halt?
- This is a decision problem but it is not in NP.
- Any NP-complete problem can be reduced to this one.

The Class NP-complete

- If a problem is in NP and also in NP-Hard, then it is an NP complete problem.
- If we know a single problem in NP-Complete that helps when we are asked to prove some other problem is NP-Complete
- Assume problem P is NP Complete, i.e. all NP problems are reducible to this problem P.
- Now given a different problem P', if we show P reducible to P', then by transitivity all NP problems are reducible to P'.

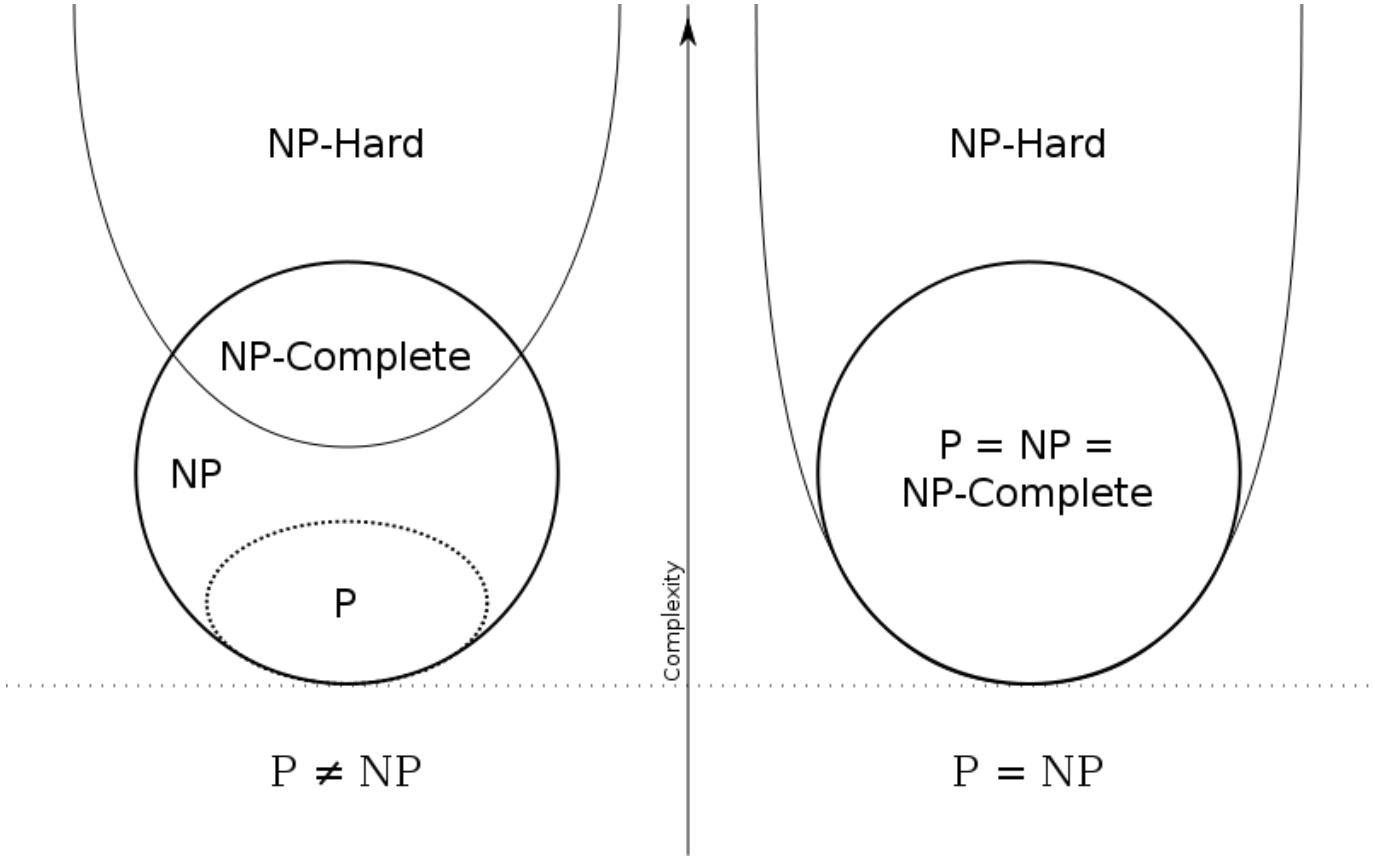
The "first" NP-complete Problem

- For the technique of reduction, it is required to have a problem already known to be NP-complete.
- The problem is the circuit-satisfiability problem.
  - Given a boolean combinational circuit composed of AND, OR, and NOT gates, determine whether there exists some set of boolean inputs to this circuit that causes its output to be 1.

Important Note

- If any NP-complete problem is polynomial-time solvable, then P = NP. Equivalently, if any problem in NP is not polynomial-time solvable, then no NP-complete problem is polynomial-time solvable.
- No polynomial-time algorithm has yet been discovered for an NP-complete problem, nor has anyone yet been able to prove that no polynomial-time algorithm can exist for any one of them.

Euler diagram



Assuming P≠NP

Assuming P=NP