Complexity Classes P and NP

MATH 3220

Supplemental Presentation

by John Aleshunas

The cure for boredom is curiosity. There is no cure for curiosity

-- *Dorothy Parker*

Computational Complexity Theory

- In computer science, **computational complexity theory** is the branch of the theory of computation that studies the resources, or *cost*, of the computation required to solve a given computational problem.
- The relative computational difficulty of computable functions is the subject matter of computational complexity.
- Complexity theory analyzes the difficulty of computational problems in terms of many different computational resources.
- Example: Mowing grass has linear complexity because it takes double the time to mow double the area. However, looking up something in a dictionary has only logarithmic complexity because a double sized dictionary only has to be opened one time more (e.g. exactly in the middle - then the problem is reduced to the half).

Source: Wikipedia (Computatoinal Complexity Theory)

Complexity Classes

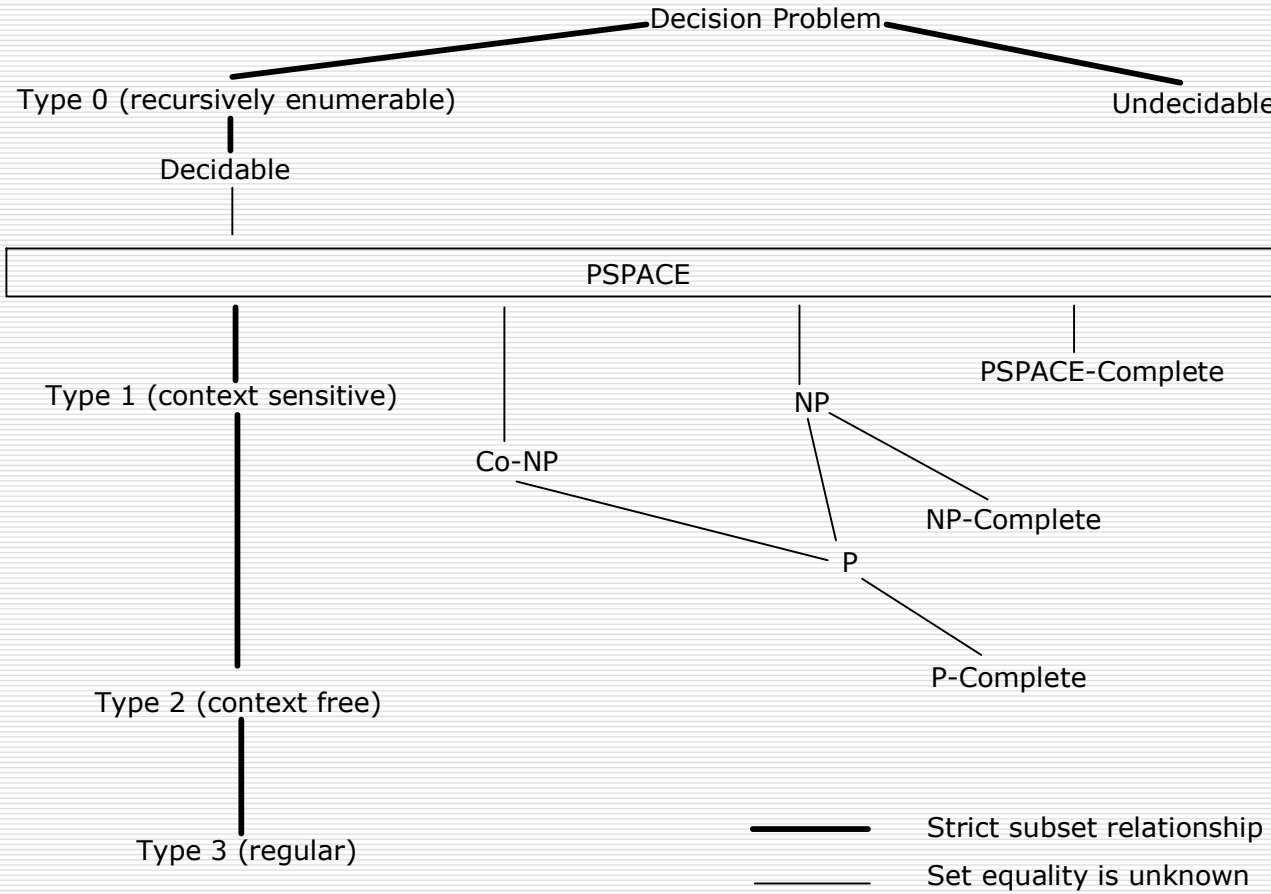
- A complexity class is the set of all of the computational problems which can be solved using a certain amount of a certain computational resource.

- The complexity class P is the set of decision problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases.

- The complexity class NP is the set of decision problems that can be solved by a non-deterministic machine in polynomial time. This class contains many problems that people would like to be able to solve effectively. All the problems in this class have the property that their solutions can be checked effectively.

Source: Wikipedia (Computatoinal Complexity Theory)

Complexity Classes (taxonomy)



Source: Wikipedia (Complexity Class)

Deterministic (Turing) Machine

- Deterministic or Turing machines are extremely basic symbol-manipulating devices which — despite their simplicity — can be adapted to simulate the logic of any computer that could possibly be constructed.
- They were described in 1936 by Alan Turing. Though they were intended to be technically feasible, Turing machines were not meant to be a practical computing technology, but a thought experiment about the limits of mechanical computation; thus they were not actually constructed.
- Studying their abstract properties yields many insights into computer science and complexity theory.
- Turing machines capture the informal notion of effective method in logic and mathematics, and provide a precise definition of an algorithm or 'mechanical procedure'.

Source: Wikipedia (Deterministic Turing Machine)

Nondeterministic (Turing) Machine

- In theoretical computer science, a non-deterministic Turing machine (NTM) is a Turing machine whose control mechanism works like a non-deterministic finite automaton.
- An ordinary (deterministic) Turing machine (DTM) has a transition function that, for a given state and symbol under the tape head, specifies three things:
  - the symbol to be written to the tape
  - the direction (left or right) in which the head should move
  - the subsequent state of the finite control
- An NTM differs in that the state and tape symbol no longer *uniquely* specify these things - many different actions may apply for the same combination of state and symbol.

Source: Wikipedia
(Nondeterministic Turing Machine)

Complexity Class P

- P is the complexity class containing decision problems which can be solved by a deterministic Turing machine using a polynomial amount of computation time, or polynomial time.
- P is often taken to be the class of computational problems which are "efficiently solvable" or "tractable".
- Problems that are solvable in theory, but cannot be solved in practice, are called *intractable*.
- There exist problems in P which are intractable in practical terms; for example, some require at least $n^{1000000}$ operations.
- P is known to contain many natural problems, including the decision versions of linear programming, calculating the greatest common divisor, and finding a maximum matching. In 2002, it was shown that the problem of determining if a number is prime is in P.

Source: Wikipedia (P (complexity))

Complexity Class NP

- In computational complexity theory, NP ("Non-deterministic Polynomial time") is the set of decision problems solvable in polynomial time on a non-deterministic Turing machine.
- It is the set of problems that can be "verified" by a deterministic Turing machine in polynomial time.
- All the problems in this class have the property that their solutions can be checked effectively.
- This class contains many problems that people would like to be able to solve effectively, including
  - the Boolean satisfiability problem (SAT)
  - the Hamiltonian path problem (special case of TSP)
  - the Vertex cover problem.

Source: Wikipedia (NP (complexity))

Complexity Class NP-Complete

- In complexity theory, the NP-complete problems are the most difficult problems in NP ("non-deterministic polynomial time") in the sense that they are the ones most likely not to be in P.
- If one could find a way to solve *any* NP-complete problem quickly (in polynomial time), then they could use that algorithm to solve *all* NP problems quickly.
- At present, all known algorithms for NP-complete problems require time that is superpolynomial in the input size.
- To solve an NP-complete problem for any nontrivial problem size, generally one of the following approaches is used:
  - Approximation
  - Probabilistic
  - Special cases
  - Heuristic

Source: Wikipedia (NP-Complete)

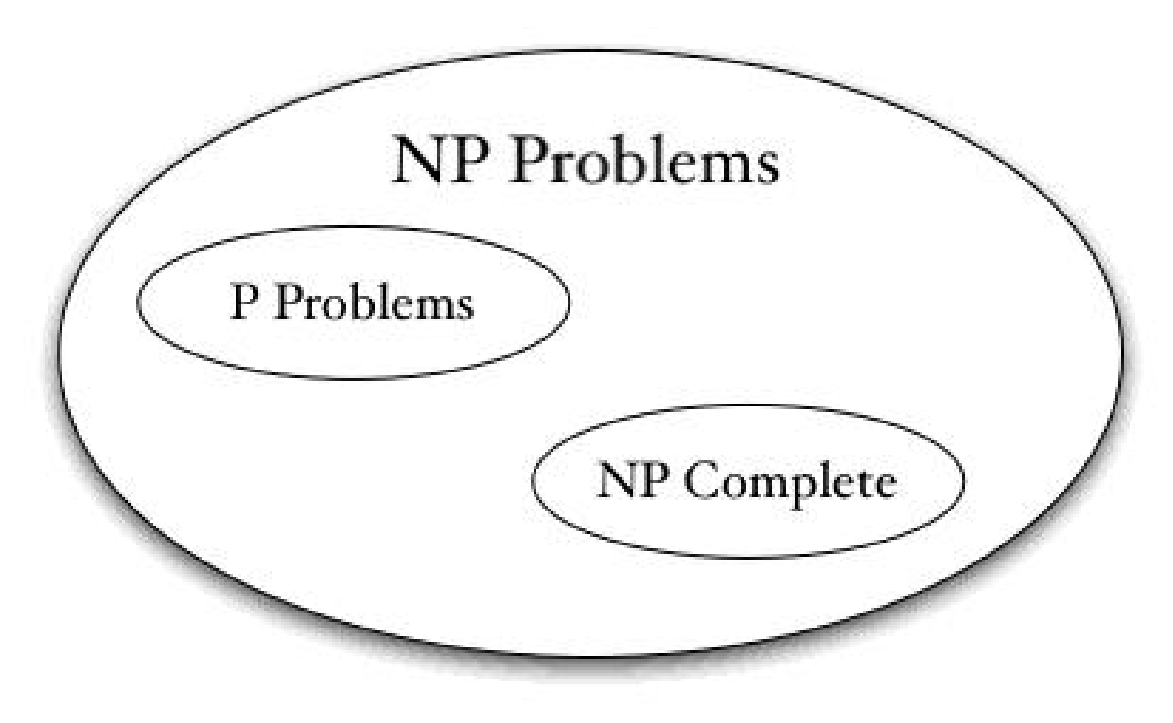
Complexity Class NP-Complete (cont)

Some well-known problems that are NP-complete are:

- Boolean satisfiability problem (SAT)
- N-puzzle
- Knapsack problem
- Hamiltonian cycle problem
- Traveling salesman problem
- Subgraph isomorphism problem
- Subset sum problem
- Clique problem
- Vertex cover problem
- Independent set problem
- Graph coloring problem
- Minesweeper

Source: Wikipedia (NP-Complete)

Complexity Classes P and NP



Source: Wikipedia (Complexity Classes P and NP)

Your Chance to be Famous

The question of whether P is the same set as NP is the most important open question in theoretical computer science. There is even a $1,000,000 prize for solving it.

Source: Wikipedia (Clay Mathematics Insitute)

I'm never going to be famous. My name will never be writ large on the roster of Those Who Do Things. I don't do any thing. Not one single thing. I used to bite my nails, but I don't even do that any more.

-- Dorothy Parker

References

- **Computational Complexity Theory**: http://en.wikipedia.org/wiki/Computational_complexity_theory
- **Complexity Class**: http://en.wikipedia.org/wiki/Complexity_class
- **Deterministic Turing Machine**: http://en.wikipedia.org/wiki/Deterministic_Turing_machine
- **Nondeterministic Turing Machine**: http://en.wikipedia.org/wiki/Non-deterministic_Turing_machine
- **P (complexity)**: http://en.wikipedia.org/wiki/P_%28complexity%29
- **NP (complexity)**: http://en.wikipedia.org/wiki/NP_%28complexity%29
- **NP-Complete**: http://en.wikipedia.org/wiki/NP-complete
- **Complexity Classes P and NP**: http://en.wikipedia.org/wiki/Complexity_classes_P_and_NP
- **Clay Mathematics Insitute**: http://en.wikipedia.org/wiki/Clay_Mathematics_Institute